The Resource Journal for IBM Micro Programmers

PROGRAMMER'S Journal

JULY/AUGUST 1988
Volume 6.4
$5.00 Canada $6.25

Opening Windows

Windowing Systems Overview
by Frank D. Greco

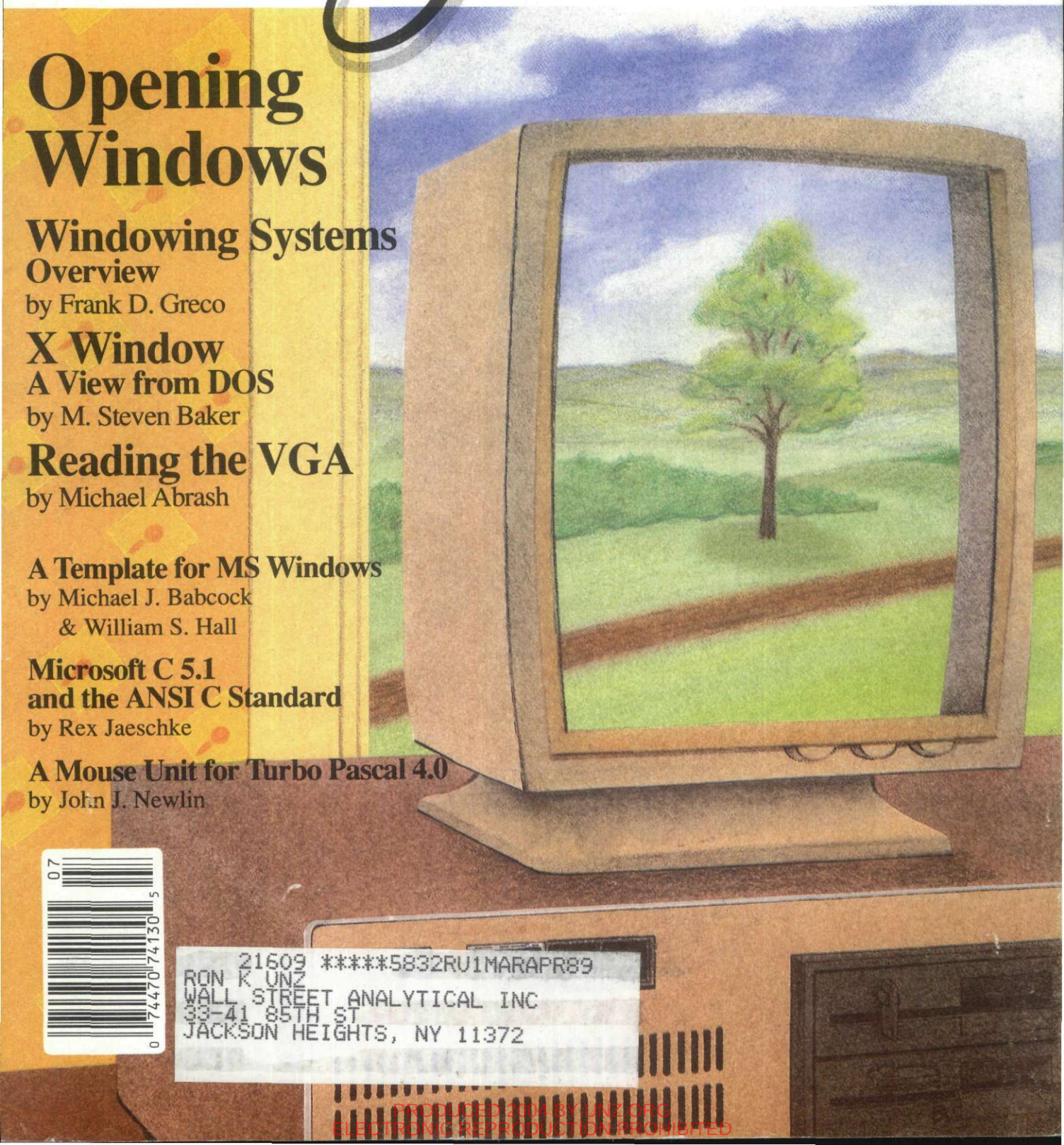
X Window A View from DOS
by M. Steven Baker

Reading the VGA
by Michael Abrash

A Template for MS Windows
by Michael J. Babcock & William S. Hall

Microsoft C 5.1 and the ANSI C Standard
by Rex Jaeschke

A Mouse Unit for Turbo Pascal 4.0
by John J. Newlin



21609 *****5832RU1MARAPR89
RON K UNZ
WALL STREET ANALYTICAL INC
33-41 85TH ST
JACKSON HEIGHTS, NY 11372

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Make your programs millions of times smarter.

More and more, programmers and workstation builders are using DESQview 2.0 as a development tool. The reason is simple. They can create powerful, multitasking solutions today for the millions of DOS PCs in use today. Solutions comparable to those promised for tomorrow by OS/2.

Some of the applications under development right now using DESQview 2.0 API Tools: CAD, Medical systems, insurance, 3270 mainframe communications, network management, real estate, typesetting, point of sale, education, commodity trading, stock trading and online voting.

The API Advantage

Programmers who take advantage of DESQview's API (Application Program Interface) get access to the powerful capabilities built into DESQview–multitasking, windowing, intertask comunications, mailboxes, shared programs, memory management, mousing, data transfer, menu-building and context sensitive help.

Bells and Whistles

A program taking advantage of the DESQview 2.0 API can spawn subtasks for performing background operations or new processes for loading and running other programs concurrently. It can schedule processing after an interval or at a certain time. It can use DESQview's intertask communications to rapidly exchange data between programs, share common code and data; or interrupt at critical events. It can use DESQview's menuing and mousing capabilities to create menus. And there's lots more it can do.

80386 Power

80386 programmers can take advantage of the 80386's protected mode for large programs, yet run on DOS and multitask in DESQview–side by side with other 80386 and DOS programs. The breakthroughs that make this possible: DOS Extenders from PharLap Software and AI Architects and DESQview support of these DOS extenders.

DESQview Developer Conference

So if you are a developer, looking to create programs with mainframe capabilities, but wanting to sell into the existing base of millions of DOS PCs, come to Quarterdeck's first DESQview API Developers Conference, August 16-18, 1988 at the Marina Beach Hotel, in Marina del Rey, California. For more information call or write us.

Come learn about the DESQview 2.0 API and 80386 DOS Extenders. Meet 80386 experts as well as those smart people who are creating DESQview 2.0 API workstations solutions.

And if you want to get a leg up before the conference, ask us about the DESQview API Tools for assembler or C programmers.

Bringing New Power to DOS. DESQview 2.0 API Toolkit.

Quarterdeck Office Systems 150 Pico Blvd.,Santa Monica, CA 90405
(213) 392 9851

Number 147 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Simply the BEST C and Pascal on AT, 386, Sun, Apollo, RT, VAX, 370

"The most rock-solid C compiler in the industry. Superb technical support and portability. Superior code generated."
Gordon Eubanks, Symantec — Q&A (386).

"It simply works, with no trouble, no chasing strange bugs, and excellent warning and error messages ... a professional product."
Robert Lerche, Bay Partners.

"For large-scale software development, the highest quality C compiler available on the market today. Pragmas are great. Quality of support is exceptional." **Randy Neilsen, Ansa—Paradox** (DOS,OS/2).

"15% smaller and 15% faster than Lattice C."
Robert Wenig, Autodesk.

"Our software is running anywhere from 30 to 50% faster than when compiled under Lattice." **David Marcus, Micronetics.**

"We switched from Lattice due to a 10% reduction in code size. The compiler is very stable." **Lee Lorenzen, Ventura Software — Ventura Publisher, marketted by Xerox Corp.**

"Best quality emitted code by any compiler I've encountered. Often amazing." **Bill Ferguson, Fox Software — FoxBase (386).**

"Messages sometimes pointed out type mismatches, incorrect-length argument lists, and uninitialized variables that had been undetected for years [in 4.x bsd]." **Larry Breed, IBM ACIS [RT PC].**

"Diagnostics turned up bugs missed by other compilers. Rapid bug fixes by technical support, someone who knew what he was talking about. 80386 code is well optimized."
Tim Addison, Logistics Data Systems.

"386 protected mode support is fantastic, especially the access to large amounts of memory. It's mainframe compute power on a PC." **Dan Eggleston, Viewlogic.**

"The preprocessor supplied with Professional Pascal is quite useful. The code quality and control over segmentation and memory models are superior to MS Pascal." **Bob Wallace, QuickSoft.**

Check Out These Reviews

- **High C ™:**

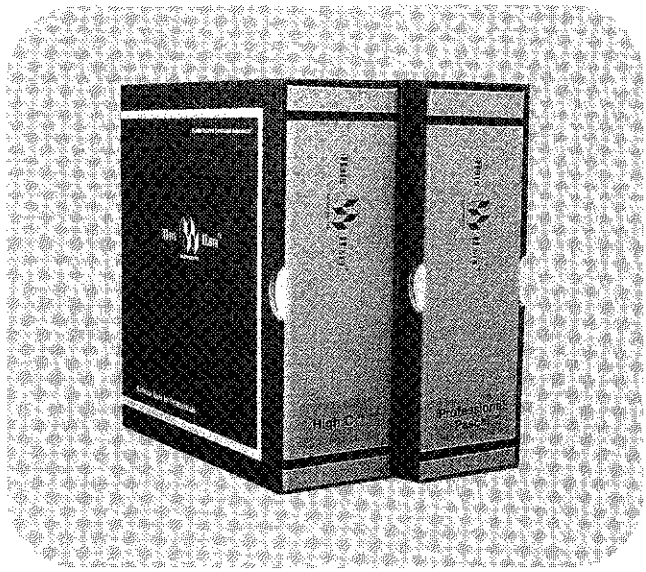
Computer Language	February 1986, '87	
Dr. Dobb's Journal	August 1986	
PC Magazine	Jan. 27, 1987	(80386 version)
Dr. Dobb's Journal	July 1987	(80386 version)
BYTE Magazine	November 1987	(80386 version)

- **Professional Pascal ™:**

PC Magazine	Dec. 29, 1985	
Computer Language	May 1986	
PC Tech Journal	July 1986	
Journal of Pascal, Ada, & Modula-2	Nov.-Dec. 1986	
BYTE Magazine	Dec. '86, June '87	(80386 version)

Why MetaWare compilers

- They are specifically designed for serious software developers.
- They are reliable and robust: they don't break at every turn.
- Their generated code is the best, or near best, on each architecture.
- Their superior diagnostic messages help you produce better products more quickly.
- Your source can be ported with ease to the most popular systems.
- You can link mixed-language modules from our compilers, others
- You can benefit from high-level, personal technical support.
- You can take advantage of the latest ANSI C extensions, and/or extensive Pascal extensions. **High C** has been tracking the ANSI Standard for two years; **Professional Pascal** will soon have a VS Pascal compatibility switch and several Apollo Pascal ext'ns.
- You can take advantage of the latest 387 and Weitek 1167 support — we have the only compilers with Weitek real mode support.



Power Tools for Power Users

Ashton-Tate: dBase III Plus, MultiMate; Autodesk: AUTOCAD, AUTOSKETCH (8087, '387, Weitek); Boeing Computer Services (Sun); CASE Technology (Sun); CAD/CAM giant Daisy ('86, '386, VAX); Deloitte Haskins & Sells; Digital Research: FlexOS; GE; IBM: 4.3/RT, 4680 OS; Lifetree Software (Pascal): Volkswriter Deluxe, GEM-Write; Lugaru: Epsilon; NYU: Ada-Ed cmplr; Semantec: Q&A; Sky Computers; ... (Product names are trademarks of the companies indicated.)

Industrial-Strength

MetaWare C and Pascal compilers are designed for professional software developers. These tools are loaded with options to control them for special purposes. You can adjust the space-time trade-off in code quality. You can adjust external naming conventions to agree with linkers and operating systems. You can specify segment names for segmented architectures, and to help place code or data in particular places for embedded applications. You can select from five memory models for the 8086 family. And on and on.

A Partial List of Optimizations

Common subexpression and dead-code elimination, retention and reuse of register contents, jump-instruction size minimization, tail merging (cross jumping), constant folding, short-circuit evaluation of Boolean expressions, strength reductions, fast procedure calls, automatic mapping of variables to registers (where advantageous), ...

"Platform" — Code Quality

Sun,Apollo,SGI	— 18%, 3%, 26% > resident compiler (Dhrystone).
PC: DOS, OS/2	— 3-10% > Microsoft C; 30% > MS Pascal, LatticeC.
386 32-bit DOS	— no competitors, since November, 1986.
286, 386 UNIX	— 66% better than pcc (Dhrystone, 386).
VAX VMS	— ≈ DEC's excellent C and Pascal; better features.
VAX Ultrix	— 19% > pcc (Dhrystone); much > Berkeley Pascal.
RT PC/4.3bsd	— 89% > the original port of pcc (Dhrystone).
370 CMS,UNIX	— much better than any C, and VS Pascal.
AMD 29000	— >40,000 Dhrystones! Available in Q2, cross.

(408) 429-6382, telex 493-0679. ***Since 1979.***

MetaWare™ INCORPORATED

903 Pacific Avenue, Suite 201 • Santa Cruz, CA 95060-4429

The Clear Choice for Large Programming Projects — PC Tech J.

© 1987 MetaWare Incorporated. MetaWare, High C, Professional Pascal, and DOS Helper are trademarks of MetaWare Incorporated. Others and their owners are duely respected.

Number 45 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal for IBM Micro Programmers

PROGRAMMER'S Journal

Volume 6.4
July/August 1988

Contents

4 Notes from the Editors

6 Letters
Mark Brand, Adrian B. Crum, Chuck Gilmore, Peter Gustafson, Graham Pearson, Richard Sanders, Keith Spitz, Tom Swan

12 Consulting: Job-shopping—Almost Consulting *by Oleg Panczenko*
What is a job-shopper, how do you become one, and what are the job prospects?

18 Redirections: Windowing Systems Overview *by Frank D. Greco*
MS Windows/Presentation Manager, GEM, X Windows, Sun NeWS, Open Look, the Apple Macintosh

22 On Graphics: Reading VGA Memory *by Michael Abrash*
The two VGA Read modes dissected

32 On Graphics: X Window—A View from DOS *by M. Steven Baker*
An initial look at the newest release, X11R2

40 C: A Digital Clock for Microsoft Windows *by Michael J. Babcock & William S. Hall*
A template for writing a first Windows application—help from some undocumented information

60 C: Windows within Windows *by John Navas II*
Using memory mapped video with TopView, DESQview, and Microsoft Windows

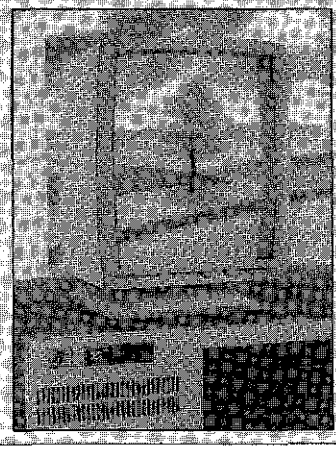
62 C: Microsoft C 5.1 and the ANSI Standard *by Rex Jaeschke*
Microsoft's and ANSI's latest versions

66 Pascal: Not a Creature Was Stirring *by John J. Newlin*
A slick mouse unit for Turbo Pascal 4.0

77 Miscellaneous

83 PJ Billboard

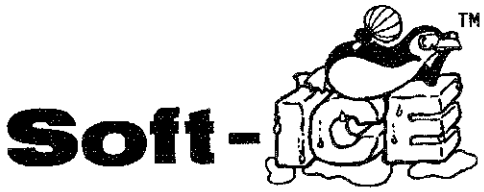
84 Advertiser Index



Will windowing ever open up on an idyll? Artist Ralph Bentley reminds us Oregonians there's good reason to endure the winter rains. Oregonians get their payoff. May there be an analogy in there somewhere, a multitasking graphical dream come true.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SERIOUS DEBUGGING AT A REASONABLE PRICE



All the speed and power of a hardware-assisted debugger at a software price

Hardware-level break points
REAL-TIME break points on memory locations, memory ranges, execution, I/O ports, hardware and software interrupts. More powerful break points than ANY software-only debugger on the market. Soft-ICE gives you the power of an in-circuit emulator on your desk.

Break out of hung programs
With a keystroke - no external switch necessary. Even with interrupts disabled.

Breaks the 640K barrier
Soft-ICE uses ZERO bytes of memory in the first 1MB of address space. This is especially useful for those subtle bugs that change when the starting address of your code changes. With Soft-ICE your code executes at the same address whether the debugger is loaded or not.

Works with your favorite debugger
Soft-ICE can be used as a stand-alone debugger or it can add its powerful break points to the software debugger you already use. You can continue to use your favorite debugger until you require Soft-ICE. Simply pop up the Soft-ICE window to set powerful real-time break points. When a break point is reached, your debugger will be activated.

Solve tough systems problems too
Soft-ICE is ideal for debugging TSRs, interrupt handlers, self booting programs, DOS loadable device drivers, non-DOS operating systems, and debugging within DOS & BIOS. Soft-ICE is also great for firmware development because Soft-ICE's break points work in ROM.

How Soft-ICE Works
Soft-ICE uses the power of the 80386 to surround your program in a virtual machine. This gives you complete control of the DOS environment, while Soft-ICE runs safely in protected mode. Soft-ICE uses 80386 protected mode features, such as paging, I/O privilege level, and break point registers, to provide real-time hardware-level break points.

"Soft-ICE is a product any MS-DOS developer serious enough to own a 386 machine should have."
Dr. Dobb's Journal -- May 1988

MagicCV $199
Soft-ICE $386
Buy Both and **Save $86**

CALL TODAY
(603) 888 - 2386

30 day money-back guarantee
Visa and Master Card accepted

NU-MEGA TECHNOLOGIES
PO Box 7607
Nashua, NH 03060-7607

Both require 80386 AT compatible or IBM PS/2 Model 80.
MagicCV requires at least 384K of extended memory.
CodeView is a trademark of Microsoft Corporation.

NEW! NEW! NEW! NEW!

RUN CODEVIEW IN ONLY 8K

CodeView is a great integrated debugger, but it uses over 200K of conventional memory. MagicCV uses advanced features of the 80386 microprocessor to load CodeView and symbols in extended memory. This allows MagicCV to run CodeView using less than 8K of conventional memory on your 80386 PC.

Don't let 640K be your limit!
If you are closing in on the 640K limit and would like the power of CodeView, MagicCV is for you.

Don't let the debugger hide the bug!
Even if you're not closing in on the 640K limit, running CodeView with MagicCV makes your debugging environment much closer to the end user's program environment. You can use CodeView to locate subtle bugs that only occur when there is plenty of free memory, or those difficult bugs that only occur when your program is running with a couple of TSRs loaded.

How MagicCV works
MagicCV uses the 80386 to create a separate virtual machine for CodeView. MagicCV uses between 4K & 8K of conventional memory as a bridge between the DOS environment and CodeView.

MagicCV is easy to use
If you are a CodeView user, you already know how to use MagicCV too. Just type MCV instead of CV; everything else is automatic.

Save $86

MagicCV with Soft-ICE
Using Soft-ICE with CodeView gives you the features necessary for professional level systems debugging. MagicCV and Soft-ICE can work in concert with CodeView to provide the most powerful debugging platform you will find anywhere. As an extra bonus, by ordering both MagicCV and Soft-ICE together you **save $86.**

Number 160 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal for IBM Micro Programmers
PROGRAMMER'S Journal

Editor
Erin O'Connor
Technical Editor
M. Steven Baker
Contributing Editors
Michael Abrash, Frank D. Greco, Tom Swan

Contributing Writers
Michael Abrash
Michael J. Babcock
M. Steven Baker
Frank D. Greco
William S. Hall
Rex Jaeschke
John Navas II
John J. Newlin
Oleg Panczenko

Publisher
Elizabeth Oakley
Production
Ralph Bentley
Illustration
Jim Carpenter
Advertising
Robert Tappan
Subscriptions
Gary Hoyer
Business Manager
Bobbi Sinyard
Administrative Assistant
Lori Fanslow
Founder
Greg Estes

Business and Editorial Offices
Programmer's Journal
150 North 4th Street
Springfield, OR 97477
(503) 747-0800

Subscriptions: 1 year (6 issues) $24
($30 in Canada and $40 elsewhere)
Change of Address
Programmer's Journal
P.O. Box 30160
Eugene, OR 97403

Articles and letters in Programmer's Journal represent the views of their authors and not necessarily those of the publisher. Letters to the editor become the property of Programmer's Journal and may be edited and published.

Code and listings in Programmer's Journal appear "as is," and Programmer's Journal makes no guarantee or warranty of their quality, performance, accuracy, or suitability for any particular purpose. Programmer's Journal assumes no responsibility or liability to any person or entity with respect to any damage caused or alleged to be caused by their use.

Advertising: For ad rates, deadlines, and placement, call Robert Tappan (503/747-0800), or write to Programmer's Journal, P. O. Box 30160, Eugene, OR 97403.

Programmer's Journal is produced on Ventura Publisher, a product of Xerox Corporation.

Programmer's Journal (ISSN 0747-586) is published bi-monthly by Oakley Publishing Company. Second-class postage paid at Springfield, OR, and additional mailing offices.

© 1988 OAKLEY PUBLISHING COMPANY

NOTES from the Editors

"Inventing Edwin Land" was the attention grabbing blurb on a recent cover of the *New York Review*, the point being, of course, that the reclusive Land has been a mystery of such a high order—right up there with J.D. Salinger—that his biographers have had to invent him. They've succeeded to a degree. Two books on Land have come out recently.

Programmers are perforce inventing windows. Among proliferating windowing systems and system calls, without a standard, and within windowing systems whose parts themselves don't conform to one another, programmers are succeeding to a degree with a routine here, a template there—recipes. If the post-modern genius is for living with uncertainty, the optimistic line of argument should maintain that our present suspension among immature and inconsistent windowing systems is a splendid opportunity for innovation. Promethean words like "independent," "heuristic," "powerful," "ambitious," should roll trippingly off the tongue. And proponents of "open endedness" should regret that thirty-six companies have gotten together to ensure that the "Open Look" will turn out to be anything but.

If Frank Greco's right, the "Open-endians" don't have much to worry about. Frank takes us on a quick tour of the contending systems and concludes that the thirty-six may, perhaps regrettably, accomplish a uniform Mac-like user interface, but that we're in little imminent danger of standardization otherwise. Wading into the divergent specifics, our Technical Editor Steve Baker has managed to get the monolithic X Windows source code into the PC and has several good ideas about where to go from there. X Windows grew out of the premise that there's no particular virtue in a mandated "look and feel" standard—different types of users and different kinds of applications call for appropriate, not standard, interfaces. John Navas takes a pan-system approach to some video BIOS extensions that allow for well-behaved and fast software performance, be it under TopView, DESQview, or Windows. Michael Babcock and William Hall provide a template that lets us get started on Microsoft's Windows now, and undocumented information of strategic value shows up right away. Next issue, Hall and Babcock will demonstrate with the same project that Windows and the Presentation Manager are not sisters under the skin.

Hobgoblin—with a foolish consistency, we misnamed Richard Wilton's *Programmer's Guide to PC and PS/2 Video Systems* (Microsoft Press) throughout Volume 6.3 (May/June 1988). Michael Abrash recommends the book in this issue—and gets its name right.

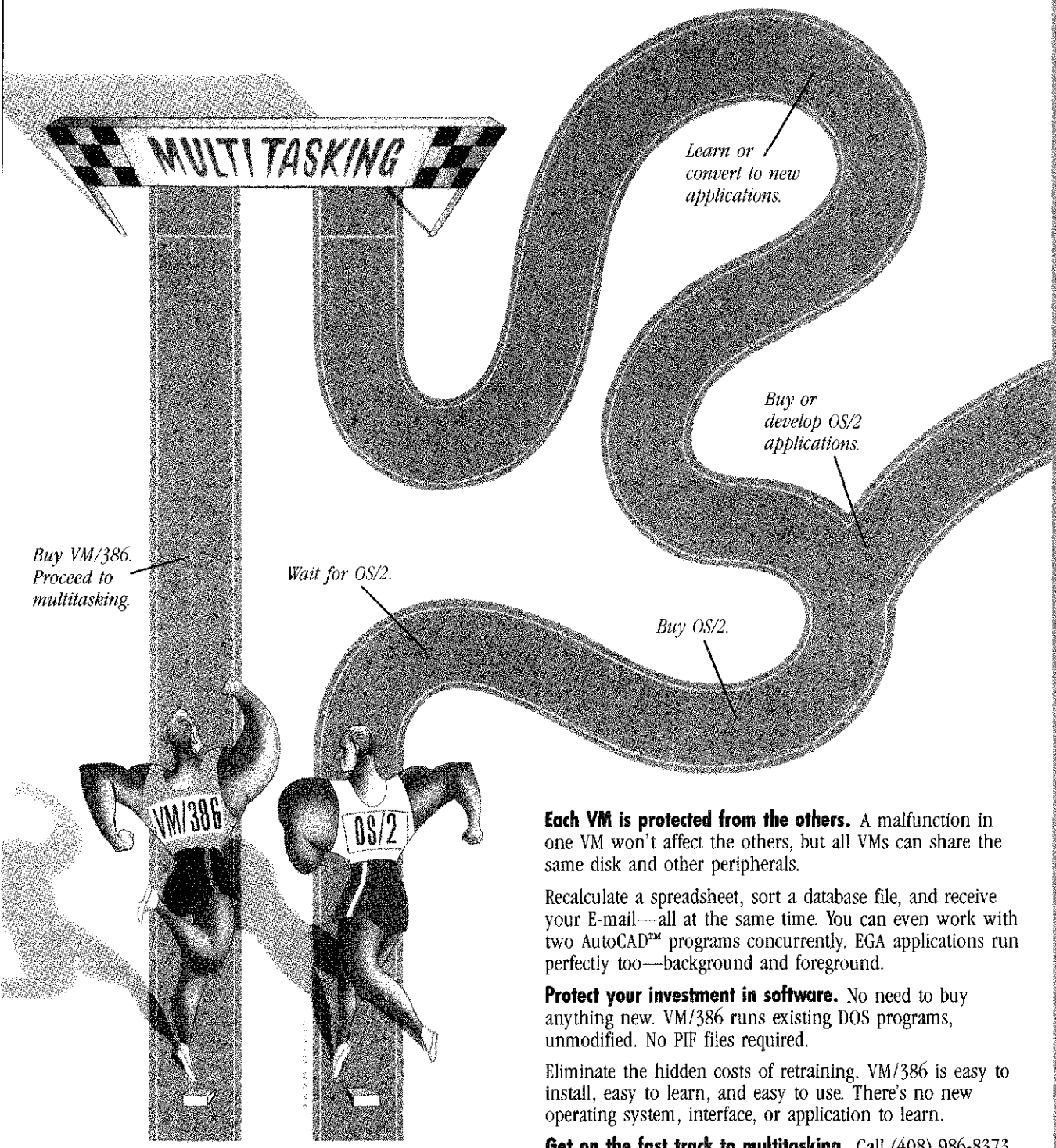
Don't let the "second edition" on the cover of Contributing Editor Tom Swan's *Mastering Turbo Pascal 4.0* mislead you. Like the compiler, the 774-page tutorial/reference book is practically brand new. Tom wrote over half the material just for the 4.0 edition and revised the rest. A code disk is available. (Hayden Books, Howard W. Sams & Co., 4300 West 62nd Street, Indianapolis, IN 46268, 317/298-5400)—*Erin*

One of my ongoing complaints about PC system software is with "undocumented features." DOS contains a number of undocumented features used by programs supplied with the system. The undocumented TSR functions are the classic, but not the only, example. I recently spent several hours poring through the Microsoft Windows SDK documentation trying to find any mention that Windows supports certain TopView system calls. These calls allow programs to be "windows aware" without a major rewrite of an application. Why wasn't this documented? And while porting a version of X Windows to the PC, I disassembled the new Microsoft Mouse driver to see the details of the hardware interface (not documented, of course). Lo and behold, I found undocumented Microsoft Mouse function calls. Would you be surprised to hear of undocumented features in OS/2 or the Presentation Manager? From my perspective, there should be no such thing as an undocumented feature.

In a similar vein, several knowledgeable folks have told me that the Intel 80386 chip is not as solid as they would like (but there are already a million of them out there now, and lots more to come). One bug, which Intel refers to as Erratum 21, can cause the machine to hang under the right circumstances when executing 80387 instructions when paging is enabled. This is a problem with the 80386 (not with the 80387), and some newer machines have been designed with a hardware workaround. Although this problem has been documented since early last fall, Intel is not supposed to correct the chip until sometime this summer. Compaq's 386 *Technical Reference* manuals do a good job of documenting their high end machines, but also provide an excellent list of 80386 bugs and possible workarounds. I highly recommend them.

I'd like to close on the issue of "compelling applications"—by this I mean new programs so attractive and powerful that they generate purchases no matter what environment they are available on. The recent announcement that Interleaf (Cambridge, MA) has ported their high end publishing software to 80386 PCs would be my current example of a compelling application for 386 DOS extenders. This high end software package was once available only on expensive minicomputers and workstations with virtual memory. Now it will run under DOS on machines with 5 megabytes of memory. In a strange quirk of fate, IBM has negotiated distribution rights for a program that isn't based on OS/2 or AIX. Interleaf developers used Phar Lap's 386|DOS Extender software and MetaWare's 386 C compiler and praised the support from both companies. Although program developers may be able to use OS/2 now, it clearly lacks compelling applications for users. Whether and when compelling applications for OS/2 arrive will determine its success.—*Steve* ♦

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



VM/386™. The Fast Track To Multitasking.

Run smart. Run efficient. Run VM/386 on your 386-based PC and start multitasking now! No detours, no waiting, no runaround.

True multitasking. VM/386 uses the virtual 8086 mode of the 80386 processor to create many Virtual Machines (VMs) in one computer.

You can load a different application into each VM. Each VM has up to 640K RAM, plus its own DOS, CONFIG.SYS, AUTOEXEC.BAT and memory-resident programs along with its applications. Tailor each VM to your needs. You have complete control.

Each VM is protected from the others. A malfunction in one VM won't affect the others, but all VMs can share the same disk and other peripherals.

Recalculate a spreadsheet, sort a database file, and receive your E-mail—all at the same time. You can even work with two AutoCAD™ programs concurrently. EGA applications run perfectly too—background and foreground.

Protect your investment in software. No need to buy anything new. VM/386 runs existing DOS programs, unmodified. No PIF files required.

Eliminate the hidden costs of retraining. VM/386 is easy to install, easy to learn, and easy to use. There's no new operating system, interface, or application to learn.

Get on the fast track to multitasking. Call (408) 986-8373 for more information or to order VM/386. Everything else is just the runaround.

System Requirements

80386-based computer such as IBM® PS/2™ Model 80 or COMPAQ® DESKPRO 386® or 80286-based computer with Intel® Inboard™ 386/AT.

One 1.2 Mb (5¼") or one 3½" microfloppy disk drive.

One hard disk drive.

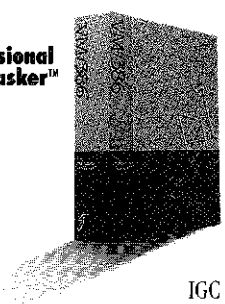
DOS 3.0 or later.

2 Mb RAM recommended.

Supports monochrome, CGA, EGA, VGA, and Hercules™ monitors.

Not copy protected.

The Professional MultiTasker™



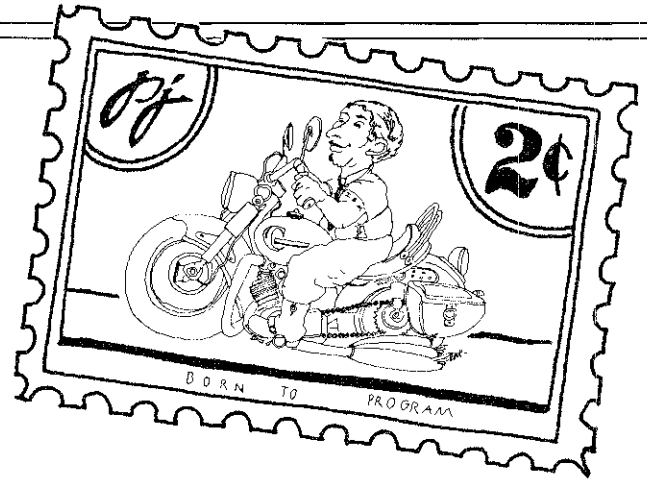
IGC
4800 Great America Parkway
Santa Clara, CA 95054
(408) 986-8373

VM/386 is a trademark of IGC. IBM is a registered trademark of International Business Machines Corporation. COMPAQ and DESKPRO 386 are registered trademarks of Compaq Computer Corporation. Intel is a registered trademark and Inboard is a trademark of Intel Corporation. Hercules is a trademark of Hercules Computer Technology. AutoCAD is a trademark of Autodesk, Inc.

VM/386

IGC

PRODUCED 2005 BY INTERNET ARCHIVE
ELECTRONIC REPRODUCTION PROHIBITED



Resident Difficulties

I need some information concerning the ASM program at the end of Graham Pearson's article, "Coding Guidelines for Resident Programs," in the March/April 1987, issue (Volume 5.2).

I have not been able to get the program to operate correctly. I think the line

```
mov     ax,2509H
```

after

```
restore_int24:
```

should read

```
mov     ax,2524H
```

Also, are there other errors? Is there a way to download a working copy of the ASM file?

Richard Sanders
Longmont, CO

Graham Pearson Replies

Richard is correct; there is a typo in the assembler code accompanying the article "Coding Guidelines for Resident Programs," *PJ* 5.2, March/April 1987. The correct segment of assembler code is:

```
restore_int24:
        lds     dx,int_24_vect
                ;Restore INT 24H
        mov     ax,2524H
        int     21H
```

There are no other errors in the listing. The corrected assembler source code has been supplied to *PJ* and can be obtained on either their BBS's or their Issue Disk or from me.

Graham Pearson
Kudos Software
9111 Cadawac
Houston, TX 77074
(713) 774-6108

The "Roll Call" Interrupt

In reference to Ben Myers' letter in *PJ* 6.3 and Michael Young's response, I'd like to help clarify the "roll call" interrupt. I believe Mr. Myers is referring to interrupt 2Fh (*not* 28h), called the "multiplex interrupt" in IBM literature. (Interrupt 28h performs a very different function and is used extensively in TSR programs to tell when it's "sort of" safe to enter DOS even when DOS is already busy.)

The multiplex interrupt has been used since DOS Version 2 to handle interprocess communications in programs like DOS PRINT. In PRINT, print function requests (queue file, dequeue file, show queue list, etc.) can be passed from a transient instance of PRINT to the memory-resident PRINT via the multiplex interrupt, just as any software interrupt could be used. The advantage of the multiplex interrupt is that a protocol has been defined that allows multiple dissimilar programs to share the same interrupt vector.

Although the multiplex interrupt has been around for quite a while, the first place I saw it defined was in the IBM DOS Version 3.1 *Technical Reference* manual. In brief, the multiplex interrupt is used as follows:

- First, each multiplex handler (that is, each program processing multiplex interrupt requests) is assigned a "multiplex number" (which I choose to call a Process ID or PID). Although 256 PIDs (00h–FFh) are available, user programs are allowed to use PIDs C0h through FFh only.
- Each multiplex handler then installs itself into the multiplex interrupt chain (by first getting and saving the current interrupt 2Fh vector address and then by pointing the 2Fh vector to the new handler routine). In all probability, the multiplex handler will become a terminate-and-stay-resident program, but it certainly doesn't have to.
- A client process (that is, a program making a multiplex interrupt request) places the appropriate PID in AH, a function code (more or less multiplex handler dependent, but see below) in AL, and other information in other registers as necessary. Then interrupt 2Fh is executed.
- Upon receiving the interrupt, the multiplex handler examines the PID in AH. If AH does not contain the correct PID, the handler is expected to chain to the next multiplex handler in the chain. The function code in AL is also examined. If the function code is in the range F8h to FFh, regardless of the PID in AH, an immediate IRET is executed (God only knows why, but that's what the specification says); otherwise, the multiplex handler processes the multiplex interrupt, returning to the client with an IRET. Values may be returned to the client via registers or by any other mechanism understood by both parties.
- Function code 0 (Get Installed State) must be supported by all multiplex handlers. Upon return from a multiplex interrupt with function 0, AL is expected to contain one of three values: 0 if the requested handler is not installed but it's okay to install it, 1 if the requested handler is not installed and it's not okay to install it, or FFh if the handler is already installed.

A couple warnings:

- Some of the PIDs are already in use by third-party software. There's no telling who has what, so you may inadvertently walk on top of someone else's PID. IBM suggests that the PID be a data variable and not an equate so that it can be patched should a conflict occur.
- Some versions of DOS, notably many Version 2s, do not initialize the interrupt 2Fh vector. This affects both the multiplex handler and client processes as follows: If the multiplex handler is required to chain to the next handler, the next handler address must first be checked and, if it is zero, an IRET executed instead; if the multiplex client wishes to execute an interrupt 2Fh, it should ensure that the multiplex interrupt vector has been initialized.

The accompanying code fragments below highlight most of these points.

I use the multiplex interrupt frequently in my TSR programs. It is reliable, efficient, and, for the most part, safe. To be honest—it takes a little longer to set up and is bound to cause problems the first couple of times through, but it works, and it works well once you get the hang of it.

For more information, the IBM DOS Version 3.x *Technical Reference* manual is, as usual, an invaluable resource.

Keith Spitz
Microcomputer Systems Manager
Computing Services
CH2M Hill, Inc.
Bellevue, WA

Code follows.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Call for Extra Savings on Weekly Specials

DISCOVER PARADISE

Programmer's Paradise Gives You Superb Selection, Personal Service and Unbeatable Prices!

Welcome to Paradise. The microcomputer software source that caters to your programming needs. Discover the Many Advantages of Paradise...

- Lowest price guaranteed
- Huge inventory, latest versions
- Technical support
- Immediate shipment
- 30-day money-back guarantee*
- Knowledgeable sales staff

Over 500 brand-name products in stock—if you don't see it, call!

We'll Match Any Nationally Advertised Price.

	LIST	OURS
386 SOFTWARE		
386-TO-THE-MAX	75	66
ALSYS 386 ADA	3655	3399
DESQVIEW	130	115
FOXBASE + /386	595	459
HIGH C-386	895	829
MICROPORT DOS MERGE (2-USERS)	399	349
MICROPORT SYS V/386 (COMPLETE)	899	759
MS WINDOWS/386	195	130
NDP C OR FORTRAN-386	595	549
PC-MOS/386 (1-USER)	195	181
PHARLAP 386|ASM/LINK	495	422
SCO VP/IX (2-USERS)	495	399
SCO XENIX SYS V 386 (COMPLETE)	1595	1279
VM/386	245	182
ARTIFICIAL INTELLIGENCE		
MULISP-87 INTERPRETER	300	199
PC SCHEME	95	86
T.I. PROCEDURE CONSULTANT	495	435
TURBO PROLOG V. 2.0 *NEW*	150	109
TURBO PROLOG TOOLBOX	100	69
ASSEMBLERS/LINKERS		
ADVANTAGE DISASSEMBLER	295	279
MS MACRO ASSEMBLER	150	99
OPTASM	195	172
PLINK86PLUS	495	279
BASIC		
DB/LIB	139	121
FINALLY!	99	90
FLASH-UP	89	80
MACH 2	79	60
MS BASIC COMPILER 6.0	295	189
MS QUICKBASIC	99	69
QUICKPAK	69	60
QUICKWINDOWS W/SOURCE	99	90
TRUE BASIC	100	90
TURBO BASIC	100	69
TURBO BASIC TOOLBOXES	100	69

DBASE TOOLS

CLIPPER	695	399
D'ACTION	80	70
D'ANALYST	229	209
DBASE III PLUS	695	399
DEBUG III	195	181
EAGLE *NEW*	495	395
FOXBASE +	395	269
FOX TOOL BOX	295	CALL
GENIFER	395	282
HI-SCREEN XL	149	129
R&R	150	139
SCANALYZER	50	45
TOM RETTIG'S LIBRARY	100	80
UTILITY PROGRAMMER	295	249

OVER 100 DATABASE PRODUCTS IN STOCK

C COMPILERS/INTERPRETERS		
C-TERP	298	232
INSTANT C	495	384
LATTICE C	450	239
MICROSOFT C	450	285
QUICK C	99	69
RUN/C PROFESSIONAL	250	159
TURBO C	100	69

	LIST	OURS
C LIBRARIES		
C ASYNCH MANAGER	175	127
C TOOLS PLUS/5.0	129	101
C UTILITY LIBRARY	185	125
CXPERT	395	335
DEVELOPER'S TOOLKIT FOR C	195	155
DESQVIEW API C LIBRARY	200	CALL
ESSENTIAL COMMUNICATIONS	185	125
COMMUNICATIONS PLUS	250	199
GREENLEAF C SAMPLER	95	69
GREENLEAF COMM LIBRARY	185	125
GREENLEAF FUNCTIONS	185	125
PFORCE	395	215
RESIDENT C W/SOURCE	198	169
TIMESLICER	295	279
TURBO C TOOLS	129	101
COBOL		
MICROFOCUS COBOL/2	900	733
W/COBOL/2 TOOLSET	1800	1465
PERSONAL COBOL	149	121
MICROSOFT COBOL	700	449
OPT-TECH SORT	149	112
REALIA COBOL	995	794
W/REAL MENU	1145	899
REALCICS	995	791
RM/COBOL	950	763
RM/SCREENS	395	339
SCREENIO	400	382
DEBUGGERS		
ADVANCED TRACE-86	175	121
BREAKOUT	125	89
PERISCOPE I	345	282
PERISCOPE II	175	141
PERISCOPE III 10 MHZ	1195	979
PFIX 86 PLUS	395	215
DISK/DOS/KEYBOARD UTILITIES		
COMMAND PLUS V. 2.0	80	70
DISK OPTIMIZER	60	55
FASTBACK +	189	142
MACE UTILITIES	99	90
NORTON COMMANDER	75	56
NORTON UTILITIES ADVANCED	150	101
PC TOOLS DELUXE	80	70
VFEATURE	80	75
XTREE PRO	129	111
EDITORS		
BRIEF	195	CALL
W/DBRIEF	275	CALL
EDIX	195	169
EMACS	295	268
EPSILON	195	151
KEDIT	150	120
MKS Vi	75	66
MULTI-EDIT	99	90
NORTON EDITOR	75	70
PC/EDT +	295	269
PI EDITOR	195	165
SPF/PC	245	185
VEDIT PLUS	185	131
FILE MANAGEMENT		
BTRIEVE	245	185
XTRIEVE	245	189
REPORT OPTION	145	109

	LIST	OURS
BTRIEVE/N	595	455
XTRIEVE/N	595	459
REPORT OPTION/N	345	279
CBTREE	159	141
C-TREE	395	318
D-TREE	495	418
R-TREE	295	241
COMBINATIONS AVAILABLE	CALL	CALL
DBC III	250	172
DBC III PLUS	750	599
DB_VISTA OR DB_QUERY	195	CALL
INFORMIX PRODUCTS	CALL	CALL
XQL	795	599
FORTRAN		
LAHEY FORTRAN F77L-EM/32	895	805
LAHEY PERSONAL FORTRAN 77	95	86
MS FORTRAN	450	285
RM/FORTRAN	595	479
DIAGRAM'ER OR DOCUMENT'ER	129	115
GRAFMATIC OR PLOTMATIC	135	119
MAGUS NUMERICAL ANALYST	295	252
SPINDRIFT LIBRARY	149	135
GRAPHICS		
ADVANTAGE GRAPHICS (C)	250	229
ESSENTIAL GRAPHICS	299	229
GSS GRAPHIC DEV. TOOLKIT	495	399
HALO '88	325	229
HALO '88 (5 MICROSOFT LANG.)	595	399
METAWINDOW PLUS	275	232
METAWINDOW/PREMIUM	495	419
TURBOWINDOW/C	95	80
TURBO HALO (FOR TURBO C)	99	80
MODULA-2		
LOGITECH MODULA-2		
COMPILER PACK	99	81
DEVELOPMENT SYSTEM	249	199
TOOLKIT	169	141
SOLID B + TOOLBOX	100	89
STONYBROOK MODULA-2	195	179
MOUSE PRODUCTS		
LOGITECH HIREZ OR SERIES 2	CALL	CALL
MICROSOFT MOUSE BUS	150	99
OTHER VARIETIES	CALL	CALL
SUMMAMOUSE	119	99
OBJECT-ORIENTED PROGRAMMING		
ACTOR	495	423
ADVANTAGE C + +	495	479
PFORCE + +	395	215
SMALLTALK/V	100	85
APPLICATION PACKS	50	45
SMALLTALK/V286	200	159
OPERATING SYSTEMS		
MICROPORT 286 DOS MERGE	249	209
MICROPORT SYS V/AT	649	549
SCO XENIX SYSTEM V (COMP.)	1295	999
WENDIN-DOS	99	80
OTHER MICROPORT, SCO, WENDIN PRODUCTS	CALL	CALL
PASCAL COMPILERS		
MICROSOFT PASCAL	300	189
PASCAL-2	229	189
TURBO PASCAL	100	69
TURBO PASCAL DEV. LIB.	395	289

	LIST	OURS
TURBO PASCAL ADD-ONS		
ASCII TURBO PROG. (COMPLETE)	289	259
DOS/BIOS & MOUSE TOOLS	75	70
METRABYTE DATA ACQ. TOOLS	100	90
OVERLAY MANAGER 4.0	45	40
SCREEN SCULPTOR	125	96

FEATURED PRODUCTS

KEDIT 4.0—Now with WIDTH option for line lengths up to 2048. New selective line editing, enhanced XEDIT compatibility, directory handling, and programmability.
List: $150 **Ours: $120**

Faceit—New development tool for designing user interfaces. Automatically builds single windows and multiple-menu systems from existing .DBF and ASCII text files. Create sophisticated menu systems in minutes; pull-downs, data entry tables, pop-ups, dialog boxes, context sensitive help windows.
List: $99 **Ours: $90**

Emerald Bay—Wayne Ratliff's revolutionary data base server technology. Application independent data sharing and multi-user functionality without additional effort.
EAGLE (revol. of dBASE)
List: $495 **Ours: $395**
DEVELOPERS TOOLKIT for C
List: $495 **Ours: $395**

GSS Development Toolkit for OS/2—Supports all OS/2 features and maintains source code compatibility with the DOS Graphics Development Toolkit. Currently the only available graphics tools for OS/2 protect mode applications.
List: $695 **Ours: $410**

T-DEBUG PLUS	45	39
W/SOURCE	90	80
TURBO ANALYST	75	59
TURBO ASYNCH PLUS	129	101
TURBO GEOMETRY LIBRARY	150	135
TURBO HALO	95	80
TURBO MAGIC	99	90
TURBO PASCAL TOOLBOXES	100	69
TURBO POWER TOOLS PLUS	129	101
TURBO PROFESSIONAL 4.0	99	80
TURBO WINDOW/PASCAL	95	80
SCREENS/WINDOWS		
C-SCAPE	299	282
CURSES W/SOURCE	250	172
GREENLEAF DATA WINDOWS	295	209
HI-SCREEN XL	149	129
JYACC JAM	750	684
MICROSOFT WINDOWS	99	69
MS WINDOWS DEVELOPMENT KIT	500	319
PANEL PLUS	495	395
PANEL/QC OR /TC	129	99
SCREENSTAR W/SOURCE	198	169
TURBO POWER SCREEN	129	101
VIEW MANAGER	275	199
VITAMIN C	225	162
VC SCREEN	149	105
WINDOWS FOR DATA	295	CALL
W/SOURCE	590	CALL
ADDITIONAL PRODUCTS		
BABY/36 (RPG II)	3000	2699
CARBON COPY PLUS	195	142
CO-SESSION	195	179
DAN BRICKLIN'S DEMO PROGRAM	75	60
DEMO PROGRAM II	195	179
DB2C	299	272
EUREKA	167	119
FLOW CHARTING II	229	207
INTERACTIVE EASY FLOW *NEW*	150	125

OS/2 DEVELOPMENT TOOLS

BTRIEVE FOR OS/2	595	455
CALIFORNIA TEN PACK	99	79
EPSILON FOR OS/2	195	151
GREENLEAF DATA WINDOWS OS/2	395	279
GSS DEV TOOLKIT FOR OS/2	695	559
HELPME	99	75
KEDIT V. 4.0	175	139
MICROFOCUS COBOL/2	900	733
MICROSOFT LANGUAGES	CALL	CALL
PANEL PLUS FOR OS/2	495	395
RBASE FOR OS/2	895	CALL
WINDOWS FOR DATA FOR OS/2	495	CALL

JANUS ADA JET SET	99	90
MAGIC PC	195	179
MATHCAD	395	279
MKS RCS	189	162
MKS TOOLKIT	169	145
MS OS/2 PROG. TOOLKIT	350	229
MUMATH	300	189
NORTON GUIDES	100	75
PC-LINT	139	101
POLYMAKE	149	131
POLYTRON PVCS	CALL	CALL
PRE-C	295	159
PROTEUS *NEW*	99	89
SEIDL VERSION MANAGER	300	269
SOURCE PRINT	97	80
TREE DIAGRAMMER	77	70

Full Line of MS-DOS, OS/2, Xenix, Macintosh Products Available.

Call or Write for Latest Free Catalog!

Terms and Policies
- We honor MC, VISA, AMERICAN EXPRESS
- No surcharge on credit card or C.O.D. Prepayment by check. New York State residents add applicable sales tax. Shipping and handling $3.95 per item within the U.S., next UPS ground. Rush and international service available. Call for prevailing rates.
- Programmer's Paradise will match any current nationally advertised price with equivalent terms for the products listed in this ad.
- Prices and policies subject to change without notice.
- Hours 9AM EST – 7PM EST.
- *Mail Orders include your phone number.*
- *Ask for details. Some manufacturers will not allow returns once disk seals are broken.

Dealers and Corporate Buyers—Call for special discounts and benefits!

1-800-445-7899
In NY: 914-332-4548
Customer Service: 914-332-0869
International Orders: 914-332-4548
Telex: 510-601-7602

Programmer's Paradise™
A Division of Hudson Technologies, Inc.
42 River Street, Tarrytown, NY 10591

Number **200** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

If you can find better sort/merge software, buy it!

Sortex:® A flexible, general purpose sort/merge facility which provides you with the ultimate in performance and reliability

- Runs from the DOS prompt, batch file, or applications written in Assembler, BASIC, COBOL, FORTRAN, Pascal, or C.
- Accepts your requirements through the full screen menu/panel facility, control file, or command line.
- Processes most of the commonly used file and data types.
- Produces output files with key, address, index, tag, or full records.
- Imposes no practical limitations. Files with four billion characters, records & fields with thirty two thousand characters each, and unlimited number of keys can be handled by Sortex.
- Allows records to be chosen selectively for processing.
- Sorts your records in ascending/descending order on any number of keys.
- Merges your sorted record sequences into one sequence on any number of keys.
- Preserves the input order of records on the output file if all their keys are identical.
- Allows keys to overlap, or be contained within other keys. They need not be contiguous.
- Processes files that contain both standard and nonstandard delimiters.

30-Day Money-Back Guarantee
Only $149.95

CORPORATION
1915 W. Orangewood Avenue
Orange, California 92668
Tel.: 714 978 0480

Number 171 On the Reader Service Card

Code from Letters—"Interrupt" by Keith Spitz

```
;*****************************************************************************
;
; This program fragment shows how the multiplex interrupt can be used to
; provide reliable interprocess communications in an MS-DOS environment.
; A single segment is assumed (i.e. a .COM file) throughout this fragment,
; so adjustments will be necessary in a multiple segment situation.
;
; Keith Spitz / CH2M HILL Inc. / May 4, 1988
;
;*****************************************************************************

;
; Define the segmentation.
;
CODE            SEGMENT                         ; A single segment.
                ASSUME  CS:CODE , DS:CODE       ; Everything goes in it.

;
; Program Data.
;
MyPIDHeader     DB      'MYPID'                 ; Helps me find MYPID if
                                                ; patching is necessary.
MyPID           DB      0C0h                    ; Assigned multiplex number.
MPXAddr         DD      ?                       ; Address of original multi-
                                                ; plex interrupt vector.

;
; See if this process is already installed.  If so, this program becomes a
; client.  If not, this program installs itself and becomes the multiplex
; handler.
;
CheckInstalled: mov     ah,MyPID                ; PID in AH.
                mov     al,0                    ; Get Installed State function.
                call    MPXInt                  ; Do the interrupt.
                cmp     al,0                    ; Not there but OK to install?
                je      DoInstall               ; Yes - become handler.
                cmp     al,1                    ; Not there and NOT OK to inst?
                je      CantInstall             ; Yes - bomb out.
                cmp     al,0FFh                 ; Already installed?
                je      AlreadyInstalled        ; Yes - become client.
                jmp     Hmmmm                   ; Hmmm.  Bomb out, I guess.

;
; At this point, the program needs to become the handler process.  It installs
; itself into the multiplex interrupt chain and then (presumably) becomes
; memory resident.
;
DoInstall:      mov     ah,35h                  ; Get interrupt vector func.
                mov     al,2Fh                  ; For the multiplex interrupt.
                int     21h                     ; Get vector.
                mov     WORD PTR MPXAddr[0],bx  ; Save offset of vector.
                mov     WORD PTR MPXAddr[2],es  ; Save segment of vector.
                mov     ah,25h                  ; Set interrupt vector func.
                mov     al,2Fh                  ; For the multiplex interrupt.
                mov     dx,OFFSET MPXHandler    ; Point at handler routine.
                int     21h                     ; Set new vector.
                ...                             ; Off to do whatever...

;
; If the program gets here, the program has decided it is a client process.
; It will perform a couple of multiplex interupt requests and then quit
; (or whatever).
;
AlreadyInstalled:mov    ah,MyPID                ; PID in AH.
                mov     al,<some function>      ; Some process func in AL.
                call    MPXInt                  ; Do the function.
                ...                             ; ... Do other things ...
                mov     ah,MyPID                ; PID in AH again.
                mov     al,<another function>   ; Another function in AL.
                call    MPXInt                  ; Do the second function.
                ...                             ; ... Do more other things ...

;
; This is the multiplex interrupt handler.  It will handle only those
; requests appropriate for the current process and pass on requests not
; for this process.  Note that DS is probably not pointing to the right
; place at the onset.
;
MPXHandler      PROC    FAR                     ; This MUST be a FAR routine.
                cmp     ah,CS: MyPID            ; Is this for us?
                jne     MPXHandNext             ; No - chain to next handler.
                cmp     al,0F8h                 ; Reserved function?
                jae     MPXHandRet              ; Yes - return immediately.
                cmp     al,0                    ; Get install state function?
                je      MPXHandFunc0            ; Yes - handle it.
                ...                             ; Check for other funcs here.
                mov     al,<Invalid function>   ;Invalid function.
                jmp     MPXHandRet              ; All done.
; Handle the 'Get Install State' function.
MPXHandFunc0:   mov     al,0FFh                 ; Yes, we are installed.
                jmp     MPXHandRet              ; All done.
; Chain to the next handler, if there is one.
MPXHandNext:    mov     ax,CS: WORD PTR MPXAddr[0] ; Offset of next in chain.
                or      ax,CS: WORD PTR MPXAddr[2] ; OR in segment.
                je      MPXHandRet              ; If 0, no one to chain to.
                jmp     CS: MPXAddr             ; Else chain to next handler.
; Return from the interrupt.
MPXHandRet:     iret                            ; All done.
MPXHandler      ENDP                            ; End of handler procedure.

;
; This routine is used to perform multiplex interrupts.  Direct INT 2Fh's
; should not be called since it is possible that the 2Fh interrupt vector
; has not been initialized in some versions of DOS.
;
MPXInt          PROC    NEAR                    ; Start of procedure.
                push    ax                      ; Save AX, BX, and ES ...
                push    bx                      ; ... while they are used ...
```


Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Spitz Code, Continued

```
                push    es                      ; ... to see if INT 2Fh is OK.
                mov     ah,35h                  ; Get interrupt vector func.
                mov     al,2Fh                  ; For the multiplex interrupt.
                int     21h                     ; Get vector.
                mov     ax,es                   ; Get the vector segment.
                or      ax,bx                   ; Or in the offset.
                pop     es                      ; Restore registers now...
                pop     bx                      ; ... (leaving flags alone ...
                pop     ax                      ; ... for later testing).
                jz      MPXIntRet               ; If addr was 0, no vector!
                int     2Fh                     ; Otherwise OK to do INT 2Fh.
MPXIntRet:      ret                             ; All done.
MPXInt          ENDP                            ; End of procedure.

CODE            ENDS
```

Two Questions

In *PJ* 6.3 (May/June 1988), you answer Konrad Wiley's letter by referring to your BBS. This sounds interesting. What is it? How can I get on it?

The second question's more technical: In house we have several Wang minicomputers and a variety of IBM PC clones. We have been printing Wang documents on PC printers by sending them asynchronously, in background, to a PC. We used a hardware interrupt to effect the operation.

We would like to do the same sort of operation, but printing off the Wang minicomputer using a parallel port and inputting the data on the PC on its parallel port. I've heard that packages exist that do this. Is there any way of using a parallel printer port as an input port? If there is, is there anything written up on how to do it using Assembly Language?

Mark Brand
Vice President
Securities Information Services, Ltd.
Passaic, NJ

You can get any of the code that appears in an issue of the magazine on any of the three PJ *bulletin board services (see the PJ Billboad page), or you can order an Issue Disk (see the order card).*

The IBM PS/2s can read and write data through their parallel ports. IBM still sells a cable and software to let you download from older PCs to the PS/2 using the parallel ports, but this kit won't allow sending to the PC. We also put your second question to the expertise of our neighbor to the east Dave Thompson, Editor of Micro Cornucopia *in Bend. Dave wasn't encouraging. Most parallel ports are hardwired for receiving data, and converting a parallel port to input data may involve major surgery depending on your hardware. Dave added that people like serial ports because they're bi-directional.—Ed.*

Reply to Dennis Reinhardt

I'd like to reply to Dennis Reinhardt's letter on OS/2 and PS/2 in your 6.3 issue. We do all of our development on an IBM PS/2 Model 80 and an IBM AT. Our product goes out on 5.25-inch floppies just as yours does. We have had only two requests for the 3.5-inch medium just as you have. This does *not* mean that people are staying away from PS/2 systems. Two kinds of products available for PS/2 machines will remedy the 3.5-inch disk problem: (1) external 5.25-inch floppy disk drives—now available for high density diskettes as well as for standard double-sided diskettes—the high density version is available for under $200; (2) "bridge" programs that allow the transfer of files via parallel or serial cable from one PC to the next (for example, from an XT or AT to a PS/2). These are available for about $60.

I, as well as multitudes of other PS/2 owners have one or the other of these two options (I have both). It doesn't make sense to buy a powerful computer system such as the PS/2 and not be able to read 5.25-inch diskettes. Most people who buy PS/2s are also buying external disk drives. One manufacturer of these drives known personally to me has trouble keeping up with the demand for them. That people aren't asking for our software on the 3.5-inch medium doesn't mean that they don't have PS/2s. A 5.25-inch diskette is preferable because it can be read by all machines as well as by a PS/2 with an external floppy drive.

As for OS/2, we're developing software for that right now. We can't afford to sit back and see if the market will want OS/2. If you'll recall, back in the days when CP/M was the dominant operating system for microcomputers everyone thought it was there to stay. When IBM first introduced the PC in 1981, nobody believed DOS would prevail over CP/M. After all, thousands of programs were available under CP/M, and almost nothing was available for DOS. The same thing is happening today—OS/2 is available now, applications for it are starting to show up, and we have the benefit of running DOS programs under it when we need to. A *true* multitasking operating system with virtual storage, micro-channel, and the ability to handle 16 Mb of RAM on the system is apples compared to the single-user DOS system oranges. Everyone has problems with "resident" programs running under DOS—Program W won't run right with Program X, which clobbers Program Y, which doesn't leave enough available memory for Program Z's operation! With OS/2, these problems won't exist—everything will run together without our having to worry about whether Program X modified the same interrupt vector that Program Y did.

So yes, we too could stand still with our AT and DOS and a "wait-and-see" attitude about whether the market will turn to PS/2s and OS/2. But what if we waited and saw that the market *did* take this turn—by that time it would be too late because the market would be overflowing with OS/2 applications and our program would be just another competitor. If IBM and Microsoft are willing to take the gamble, I am, too. Does anyone else agree?

Chuck Gilmore
Gilmore Systems
Beverly Hills, CA

Efficient Jump Tables

I'd like to clarify one point about the assembly language program that Tom Swan provided in his article "A CASE of the Jumps" in Volume 6.3. Mr. Swan stated that " . . . there is one main disadvantage to using jump tables. If your case selector values are widely separated, you must insert dummy offsets into the table." This is true only for the code that Mr. Swan provides; there are more efficient ways to implement jump tables. I recommend the following changes to Mr. Swan's Listing 2:

1. Insert a table to valid selector values in the data segment.

2. Add the offset of the endcase procedure to the offset table.

3. Initialize ES at the program start.

4. Use the 8088 "SCASB" opcode to locate the correct procedure.

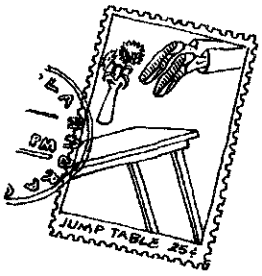
This is how the accompanying routine works: DX is initialized with the offset of the error message (as in the original listing), and DI is initialized with the offset of the character table. The processor scans through the table and stops when a match is found for the character contained in AL (DI will point to the matching character). If no match is found, DI points to the character past the end of the table. The offset of the character table is subtracted from DI, yielding a value equal to the offset of the matching character within the table. This value is multiplied times 2 to produce the corresponding word sized offset within the jump table. If no matching character was found, the result in DI will be the last offset in the jump table, which, in this case, is the endcase procedure.

Continued . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

This method will always execute the endcase procedure when an invalid character is placed in AL, regardless of how widespread the selector values may be. This is accomplished without adding unnecessary offsets to the jump table.

Adrian B. Crum
La Habra, CA 90631



Tom Swan Replies
While Mr. Crum has an interesting idea, I disagree that his solution is "more efficient." In good jump table programming, the data are used at random in calculating indexes into the table of addresses, which point to the routines that are to be executed for particular data elements. A classic jump table or Pascal CASE statement (not Turbo's, though) operates at the identical speed for all possible input values.

```
------------------------------------------------------------
        .DATA

;-----  Command  character list

chars   DB      "0123"
lchars  EQU     $-chars             ; length of command character list

;-----  Jump (case) table of addresses

table   DW      case0,case1,case2,case3,endcase

;-----  Various strings

etc...

------------------------------------------------------------------
        .CODE
start:
        mov     ax,@data            ; initialize DS & ES
        mov     ds,ax
        mov     es,ax

etc...

------------------------------------------------------------------

;-----  Prepare DI equal to the case statement number times2,
       indexing the 2-byte addresses in the jump table.

        mov     dx,OFFSET errmsg    ; Prepare for possible error
        mov     di,OFFSET chars     ; Point to command character table
        mov     cx,lchars+1         ; Load length of character table
        cld                         ; Scan forward
        repne   scasb               ; Find the command character
        sub     di,OFFSET chars+1   ; Point to procedure
        shl     di,1                ; Convert to word address
        jmp     table[di]           ; Jump to case 0,1,2,3, or endcase

etc...

------------------------------------------------------------------
```

Continued . . .

With Prospero PC Pascal and PC Fortran you never lose your *train...*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The problem with Mr Crum's code is the REPNE SCASB instruction, which scans a string of test characters. Suppose this string contains the letters ACFHKMPRTWZ. Obviously, scanning for the letter A takes the least time while scanning for Z requires examining 10 previous characters. True, such a scan will not take a lot of time, but it will take some time. Inputting a zillion Zs will cause the program to run more slowly than inputting an abundance of As.

This reintroduces the very problem I was trying to solve in the article. Because Turbo Pascal compiles CASE statements to a series of compare and jump instructions, the higher labeled statements take less time to execute than the lower ones. Using a jump table dramatically improves execution time by giving all input values a fair shake. Inserting dummy entries for unused labels guarantees that the CASE statement will operate in the same amount of time for all input values, an important quality in programs that require detailed analysis of execution times, or in code that must be synchronized with other events.

These dummy entries waste memory, as I pointed out. To reclaim this space, you can use a solution like Mr. Crum's, sacrificing speed and evenness to save a few bytes, or you can use my solution, which gives the best running times at the expense of a little memory. As it is so often, your choice is speed or memory savings. You can't have it both ways.

Tom Swan
Lititz, PA

Controlling the Beep

I'm a relatively new reader of *PJ* and have just sent in for my subscription. So far I'm pretty impressed by the publication.

In our work we use the Compaq 386/20 as a controller for a specialized piece of hardware. We use the default beep on the computer for audible feedback. Unfortunately, the default beep has changed many times and is actually long enough in duration now to slow our other processing. We would like to continue to use the default beep (triggered by a Ctrl-G sequence written to the standard output device), but would prefer to be able to change the duration and tone permanently. We have some older installed systems that use IBM AT computers as controllers, and basically, every model has a different beep. We'd like a solution that can be used on any compatible machine if that's possible. If it's in the BIOS, that may be a problem, but we think it is possible that we could modify the ANSI.SYS driver (which we currently use as a standard practice). The beep may not seem like a big deal, but believe me, it would be very helpful to be able to control it. We know how to control the beep as a general interrupt call, but the default tone still escapes us.

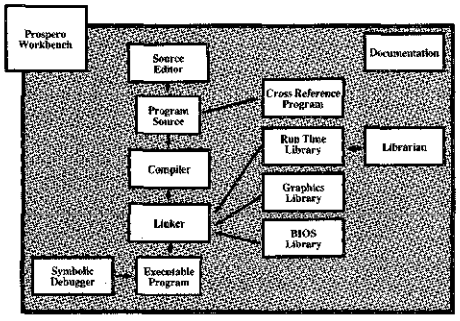
Peter C. Gustafson
W. Melbourne, FL

We've said for a long time now that what the PC needs is a good "thud." Of the approaches you might take, revising the ANSI.SYS driver seems like the weakest—it probably makes a call to the ROM BIOS. We suggest that you write a beep routine to use in your code. You can use the Beep routine from the PC or AT ROM BIOS for a guide, varying the sound, frequency, and timing until you're satisfied. In the section of the PC or AT Technical Reference *(older ones) devoted to source code for the ROM, you'll find a Beep routine. Use this code snippet in your own code. Another way to go about it is to install your own sound device driver to control the sound chip. Send messages from your code to the device driver (the same driver for all machines) to control tone (you can make the IBM thud instead of beep) and vary duration (but not, alas, volume). Or you might write a TSR that intercepts calls to the ROM BIOS for video output and checks for the bell character's being sent to the Write TTY function. Rather than invoke the ROM BIOS, invoke your own code.—Ed.* ♦

...of thought.

Introducing the world's most productive programmer's environment: The Prospero Workbench.

Whether you program in Pascal or Fortran, you'll be more productive with the Prospero Workbench. It's a fully integrated programmer's environment for Fortran. And the world's most creative environment for Pascal.



The Prospero Workbench lets you design, write, debug, test and document a program in sequence, without interruption. So, you can never lose track of a complex variable in Fortran, or get side-tracked in Pascal.

With pop down menus and four or eight editing windows, the Prospero Workbench lets you know what's happening all the time. You can read and edit files in different windows, and easily copy text from one file to another—at the press of a key.

PC Fortran and PC Pascal include the widely praised symbolic debugger PROBE. And Prospero fully implements ISO and ANSI standards. If you prepare programs on a microcomputer, and then run them on a mainframe, you know how important that is.

Prospero Software is used by more than 20,000 programmers around the world. Prospero also provides technical support, just in case you do get side-tracked.

Call 800-327-6730 to order or to request more information, and get your programming on track with Prospero PC Fortran and PC Pascal.

PC FORTRAN $199
PC PASCAL $149

Prospero Software
LANGUAGES FOR MICROCOMPUTER PROFESSIONALS

1-800-327-6730

100 Commercial Street, Portland, Maine 04101

Number 199 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Job-shopping

Almost Consulting

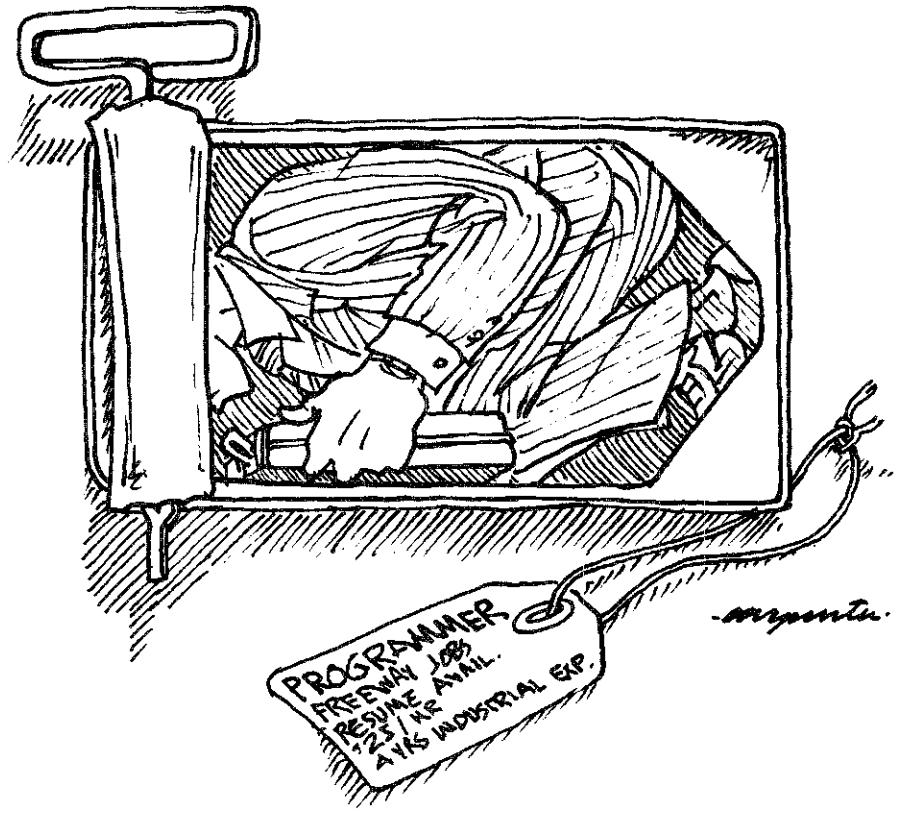
by
Oleg Panczenko

Paul Barkley's articles on consulting have long fueled readers' aspirations to go out on their own. But an employee's life is comfortable, and a consultant's responsibilities so intimidating that most don't advance beyond daydreaming. I, too, just daydreamed until fortune propelled me towards what I really wanted to do.

Two years ago I'd left a solid, if spiritless, company to work for a small start-up enterprise. I had stock options, and the company promised to be another Lotus.

The enterprise went bankrupt at the beginning of 1987.

My severance pay wasn't enough to pay for a trip around the world, but it was enough to settle all my debts. I figured this was a good time to try my hand at independent work. What kind of work can you do if you're too dumb to be a consultant but smart enough to work for yourself? Drawn by the prospect of earning $30 to $40 an hour, I decided to try my hand at *job-shopping*.



What Is Job Shopping?

As a job-shopper you're also called a freelance employee, contract employee, or technical professional. You are a master of some particular aspect of a profession for which you are highly paid. A job-shopper can go into a company on short notice, do a job, and leave. Job-shoppers work both for small companies and for big companies such as IBM, Bell Labs, Boeing, American Express, and Chemical Bank.

A number of sound reasons motivate a company to hire an expensive temporary employee rather than an inexpensive permanent one: First, it takes several weeks and quite a bit of money for a company to find a full-time employee. If a job has to be done immediately, a job-shopper can be found and hired within days. Second, the job may be a one-time service and the company have no need for a full-time employee to perform that particular service. Third, no company wants to fire people when business is slow. A company can hire contractors when times are good and let them go when times are tight without creating uncertainty or ill-will among its permanent employees.

A job-shopper trades a willingness to put up with uncertainty and to stay flexible for money. The shopper can be hired or fired at a moment's notice, and on a few day's notice may relocate to another state. Those willing to go anywhere there is work are said to prefer *road jobs*. This kind of shopper may get a weekly living allowance (the misleadingly named *per diem*) and travel pay. The client decides to hire this kind of shopper on the basis of his or her resume and perhaps a phone interview.

Those who prefer work within commuting distance can get *freeway* jobs. This is quite practical in urban areas such as Boston, Chicago, Houston, Los Angeles, San Francisco, and New York City. The rates for freeway jobs may be lower than those for road jobs. For freeway jobs, it's likely that the client will want a face-to-face interview. A freeway shopper works to establish a reputation with a set of local clients so that he or she may go from one to the next, assured of more or less constant employment.

Brokers

As a contract programmer you are a temporary hourly employee of a contract service firm, also called a *job-shop*. You announce yourself to a shop, which tries to match your skills with the requirements posted by the shop's clients. Many shops do nationwide placement: Just because the shop is local doesn't mean that it places only locally. Some shops will screen you: Their recruiters will call you in for a face-to-face technical interview or arrange one over the phone. Other shops will go on the basis of the recommendations made by your references. Some go strictly by your resume.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The shops make their money under one of two basic types of arrangements with clients. The first is the *cost-plus* arrangement, where the client pays the hourly rate paid the programmer plus a mark-up for the shop's costs and profit. If you get $20 an hour and the shop charges a 25% mark-up, the shop bills the client $25 an hour. The incentive is for the shop to get the job-shopper as high a rate as it can. The other type of arrangement is *fixed price*. The shop contracts with a client to provide programming services for a given amount per unit time. The shop's profit then depends on how little they can get away with paying you.

No matter what kind of agency you deal with, none will charge you fees. Run, do not walk, from any agency that wants money from you.

The shop, not the client, pays you. The shop deducts FICA, federal and state taxes, social security, unemployment insurance, and any other deductions from your gross earnings. The shop will either mail you your check or you may pick it up at their offices. At the end of the year the shop will mail you your form W-2.

How Much Is Your Time Worth?

I've mentioned that you may expect between $30 and $40 an hour. But these big bucks are not so big. A regular job also provides you with benefits. Some contracting firms do offer vacation pay, insurance, and a retirement plan (401K) and may even have profit sharing. But all this comes after you've worked as a contractor for a while. When you're a beginner, the only thing your employer gives you is money. You have to provide your own health insurance (look into an HMO), life insurance, and retirement fund. You can delay consideration of some of these concerns, but not for too long. It takes the better part of a life to provide enough for old age. You have to work at least ten years at a company before you become vested in their pension plan. Bearing this in mind, you have to first calculate what is the true value of your time as a regular (also called *direct*) employee before you can figure out what your time is worth. As a rough guide, use this formula: Your rate should equal your present salary times the sum of 1 plus your benefits expressed as a fraction of your salary, all divided by the product of the fraction of a year you'll be working times the number of working hours in a year (2080).

For example, if your salary is $33,000 a year and your benefits are worth 33% of your base pay and you expect to be working nine months out of the year as a contractor, you would have to earn over $28 an hour as a contractor just to break even.

The rate you're paid depends on many factors. In the Northeast and California, expect $35–$40 an hour. In the South, expect $30–$35 an hour. If you have a security clearance, you can get $40–$45 an hour from the aerospace industry in the Pacific Northwest. These rates are approximate. I've had offers as low as $20–$25 an hour. My minimum is $30 an hour. Your rate also depends on the market, your experience, the demand for your skills, the duration of the contract (short ones usually pay more), location, cost-of-living in a given area, etc. Many recruiters are friendly and willing to talk. Don't be afraid to ask them about the range of going rates, the market, and what skills are hot.

Now, under Section 1706 of the Tax Reform Act of 1986, job-shoppers are considered to be temporary hourly employees of the brokers.

Getting a Fair Day's Pay

Ask about the following when you inquire about a job:

- The duration of the assignment. Some may be as short as a month; others may last several years. Most fall between six months and a year.
- The rate for the job. Your rate of pay may be stated in hourly, daily, or weekly terms. The daily and weekly rates assume a fixed amount of time spent each day or week. Ask what that amount is. Convert the figure to rate per hour.
- Overtime and the overtime rate. Some places pay straight time for overtime, others time-and-a-half.
- The frequency of payment: weekly, fortnightly, monthly.
- Special rates for worked holidays.
- Raises if the contract is to run for a year or more. When? How much?
- Mileage allowances if the job is further than 50 miles from home.
- Travel allowances and *per diems* if you must relocate. The *per diem* typically ranges between $50 and $120 a week. The client will usually pay for one-way transportation.

1706 and Taxes

In the past, the job-shopper was an independent; the job-shop acted only as a broker. Job shoppers were self-employed and could take normal business deductions from their taxable income. Now, under Section 1706 of the Tax Reform Act of 1986, job-shoppers are considered to be temporary hourly employees of the brokers. Brokers are lobbying for a repeal of this section of the tax law. With all the legal and political goings-on, the status of Section 1706 is in flux. Ask your shop for the latest information about 1706. Mr. Michael J. Wallach, who is associated with the Independent Computer Consultants Association (ICCA), welcomes calls from anyone interested in Section 1706. Call him at (718) 317-1813. By the time you read this, the ICCA should have a service in place that will send non-members information and news about 1706 for a nominal fee to cover copying and mailing costs.

What's the Work Like?

Expect what you're used to. Some places treat you just like an employee; the pressure, or lack of it, independent of the fact that you are a contractor. Some supervisors are relaxed and informal with contractors; others apply the sit-on-their-necks method of personnel motivation.

You'll find all the pathologies you find in the direct world. Some employers will tell you to waste no time designing or documenting: You are to crank out code, period. Others are too busy to tell the man they're paying $40 an hour what to do. There are those who are pleased with whatever you propose to do yet are never satisfied with what you've done.

Risks and Failure

There is a price for knowledge, and it's going to cost you something to find out if you can make a living as a job-shopper.

There will be a period of uncertainty about the future. Be prudent—spend less money. Uncertainty and your sudden frugality may strain personal relationships.

Come up with a cut-off date. After this date, begin a parallel search, looking for both contract and direct work. Give yourself six months to find contract work. Remember that it takes from six months to a year to find a good, regular full-time job. You may have

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Eco-C88 C Compiler with Cmore Debugger

Professionals prefer the Eco-C88 C compiler for ease of use and its powerful debugging features. Our "picky flag" gives you nine levels of lint-like error checking and makes debugging easy.

"I'm very impressed with the compiler, editor, and debugger. I've tried quite a few different compilers for the PC and have given up on all of the others in favor of yours... I've gotten to the point where I download C code from a DEC VAX/VMS system just to be able to compile it with the picky flag set at 9. It finds lots of things VMS totally ignores..."
JS, Oak Ridge, TN

The Eco-C88 compiler includes:

- A full-featured C compiler with 4 memory models (up to 1 meg of code and data) plus most ANSI enhancements.
- Without a doubt, the best error checking you can get. We catch bugs the others miss, making you much more productive.
- Cmore is a full-featured source code debugger, not some stripped-down version.
- Robust standard library with over 230 useful (no "fluff") functions, many of which are System V and ANSI compatible. Full source is available for only $25.00 at time of order.
- CED, a fast, full screen, multiple-window program editor with on-line function help. You can compile, edit, and link from within CED.
- cc and mini-make utilities included that simplifies the most complex compiles.
- Users manual with over 150 program examples (not fragments) to illustrate how to use the library functions.
- Fast compiles producing fast code.

Our Guarantee: Try the Eco-C88 compiler for $99.95. Use it for 30 days and if you are not completely satisfied, simply return it for a full refund. We are confident that once you've tried Eco-C88, you'll never use anything else. Call or write today!

Orders: **1-800-952-0472**
Info: **1-317-255-6476**

Ecosoft Inc.
6413 N. College Avenue
Indianapolis, IN 46220

ECOSOFT

Number 8 on the Reader Service Card

some difficulty returning to the world of direct work. Some employers think that someone who's demonstrated a taste for the independent life can't become a "team player."

Before You Try Job-shopping

Don't think of job-shopping unless you have lots of money set aside. You should have enough money to sustain yourself for at least six months. Take unemployment benefits into account.

Before you start, get rid of debt. I was fortunate in that my severance pay was sufficient to pay off my obligations.

Easy-to-anticipate obligations are rent, utilities, loan payments; the gotchas are the easy-to-forget semi-annual or annual payments for things such as

- auto insurance
- auto repairs and maintenance
- other insurance premiums (health, renter's)
- postal-box rental

Other common payments will increase once you become a shopper:

- Phone bills—these will go up even though some shops have toll-free numbers or will accept collect calls. Assume $30–$50 a month extra.
- Postage, envelopes, copying costs. Assume $25 a month.

Some new expenses will be printing costs for resumes and the costs of using your car or train fare if you are interviewed by local shops.

Looking for Work

Looking for work is full-time work. A typical workweek starts on Sunday. Get the paper and look for agencies. On Monday mail out resumes. Get on the phone and begin calling the agencies you mailed out resumes to the previous Monday. Then make follow-up calls to agencies that received your resume the previous month.

Preparing Your Resume

A good resume is more important when you are seeking contract work than when you are looking for a direct job. Often you will get your job on the basis of only your resume—no interviews and no personal meetings.

There are books full of resume formats. In general, have your name, address, and phone number at the top, then your title (programmer, software engineer), then a paragraph describing your strengths. *Presume that this paragraph is all that the busy recruiter will glance at.* Use a telegraphic style and state the languages, operating systems, hardware, software products, databases, and utilities that you know well. Describe your skills, such as product development, documentation writing, specifications writing. Follow with your work experience: Briefly but in detail describe what you did or learned at each job. Put yourself in the place of a client. Ask "what can this guy do for me?" When you prepare a history of your contract work, include the date of the assignments, but place them in such a way that they may be unobtrusively whitened out. Some agencies want workdates—others don't. This method accommodates both.

Typeset or laser print your resume. A resume set up typographically draws a busy reader's attention to what is most important. Use boldface for key words such as languages, hardware, special knowledge (linear algebra, for example). Don't go font-happy: A clean resume will be read; a rococo resume won't be.

Don't bother with cover letters—no one is going to read them.

Typesetting costs about $40 per page. Your typeset copy can be delivered as either the original (suitable for offset printing) or a PMT copy (suitable for bond-type copiers). A PMT copy is printed on a sheet of white plastic. (Sometimes exposure to strong light, as during copying, causes the PMT page to brown. This brown film is a photosensitive residue easily rinsed off.)

A word processing service with a laser printer can produce a resume that looks almost a good as one that is typeset—for less money. Many printers offer this service. Look for the tags "Laser Typesetting" or "Desktop Publishing." Typical fees are $25 for the first page of a resume and $5 each for succeeding pages.

Note that copying costs are extra.

Don't bother with cover letters—no one is going to read them.

Where to Send Your Resume

You can find addresses of job-shops in the special publications for contract engineers, in large-city newspapers, and in the yellow pages of the phone book.

I subscribed to *Contract Engineer Weekly* (C.E. Publications, Inc., P.O. Box 97000, Kirkland, WA 98083, 206/823-2222; $45 for 52 issues, $3 for a single copy), which covers the entire contract engineering industry, not just programming or software. With your subscription you get a *Directory of Contract Service Firms*. C.E. Publications also provides services such as typesetting, printing, and mailing of resumes, which I found to be satisfactory and competitively priced. Don't buy their video *Highway to Success*—it's superficial.

In the newspaper under "Programmers," look for ads with captions such as "$600/DAY" or "$10,000/MO" or "CONSULTING," or "HI $$$/DAY," or "FREELANCE EMPLOYEES."

Look in the yellow pages under

- "Employment Agencies"—in the ads look for "Temporary Division" or "Permanent/Temporary/Part Time."

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

- "Employment Contractors/Temporary Help"—look for "Technical—Job Shop" in the ads.
- "Employment, Temporary."

Most of the agencies you find in the phone book will deal with jobs such as Accounting, Data Entry, Mail Clerking, Stock Picking, and Word processing, but a few agencies mention technical jobs in their ads.

If your employer hires job-shoppers, ask one of them for advice.

Some agencies acknowledge the receipt of your resume with a postcard or letter; others will call; about 50% won't acknowledge at all.

Follow-up Calls

Make follow-up calls—they're important. Some recruiters have little idea of what they are working with. I had to correct one recruiter's pronunciation of technical words and infer from what he stumbled through what the requirements of a particular job were.

Shops mechanically match the buzzwords in your resume with the buzzwords in the requirements submitted by the client. Often a client's requirement and your qualification fit but are described in slightly different terms—something a mechanical correlation won't match. Some qualifications are not imperative, and you may be strong in other areas: The client will accept you even though you are not an exact match. Remember, it's unlikely that clients will get exactly what they want: They have to settle for what they can get. Talk to a recruiter to find out what really matters and persuade the recruiter to send your resume for consideration. At the very least when you call, the recruiter pulls out your resume—in sight, in mind.

Keep records. I use a 3 x 5 card with the name, address, and phone number of the shop. I add the name of a contact, the dates I mailed resumes, when the shop replied (card or call), and when I placed follow-up calls.

Do You or Don't You Have a Job?

Eventually, you'll happen on a match of requirements and qualifications. You'll have a phone interview with the client. Then you'll wait for a reply. Most agencies won't call you to tell you that you've not gotten a job.

Be prepared to begin work on short notice. You may be interviewed by the shop Tuesday, by the client Wednesday, and begin work Friday!

Accept only jobs where you know the work unless the employer agrees in advance that you will spend some time learning—an assignment is not an occasion for on-the-job training.

What to Do When You Get the Job

After you've accepted the job, request a confirming letter or telegram containing (1) your hourly rate, (2) to whom you should report on the first day of work and his or her phone

FINALLY!

EASYFLOW

An on-screen flowchart processor that knows about flowcharts - not just another "screen draw" program that makes you do most of the work. EASYFLOW is a powerful full-screen graphics program dedicated to flowcharts and organization charts. With it you can quickly compose charts. More important, you can easily modify charts so they are always up to date.

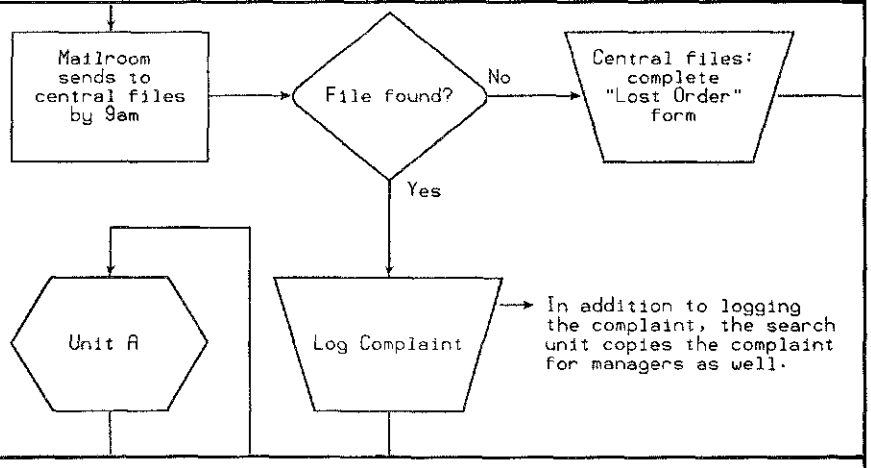
- **Automatic:** Fully automatic text centering within shapes, both horizontally and vertically. Fully automatic line routing & re-routing.
- **Fast:** Written in assembly language for speed.
- **Large:** Charts up to 417 columns wide by 225 lines high. Chart too large for your printer? EASYFLOW automatically breaks the chart up & prints it in page size pieces.
- **Standard:** All standard flowcharting shapes included. Custom shapes can be ordered.
- **User friendly:** Don't take our word for it. PC Magazine* says "EASYFLOW lives up to its name. It's hard to imagine any easier and more flexible way to produce basic and even complex flowcharts".
- **Mouse:** Optional but fully supported.
- **It prints:** On most popular matrix printers including IBM, Epson, Toshiba, HP LaserJet, LaserJet-Plus and many others.
- **It plots:** On HP7440, 7475, 7550, 7570, 7585B and compatible plotters.
- **It works:** We are contractually prevented from mentioning the name of the "big eight" accounting firm that purchased a world-wide site license, but we can tell you that they spent months evaluating all available flowcharting packages before choosing EASYFLOW.
- **Rush delivery:** Order by noon today (eastern time) and we'll have it to you by courier tomorrow**. Rush delivery charge is $15.00 (instead of $2.00) and is available only in USA & Canada.
- **Documented:** 100 page manual plus over 150 screens of context sensitive help.

EASYFLOW works on IBM PC's, IBM PS/2 and compatibles. Requires 384 K memory, DOS 2.0 or higher and an IBM CGA/EGA/VGA or Hercules monochrome compatible adapter card.

Order direct for only **$149.95** + $2.00 S&H (USA/Canada), $10.00 (foreign). Payment by M.O., cheque, VISA, Mastercard or Company PO.

* March 10, 1987 issue, page 278.

** Rush orders are shipped by Purolator Courier and normally arrive the next business day to most locations. Remote destinations take longer.


Mailroom sends to central files by 9am
File found?
No
Central files: complete "Lost Order" form
Yes
Unit A
Log Complaint
In addition to logging the complaint, the search unit copies the complaint for managers as well.

The chart fragment above was produced on an HP LaserJet-Plus and is actual size and unretouched. Publication quality charts like this can be produced using only minutes of preparation time and seconds of print time.

HavenTree Software Limited
PO Box 1093-V
Thousand Island Park, NY 13692

Order Desk: 1-800-267-0668
Information: (613) 544-6035 ext 31
Telefax(G3): (613) 544-9632

Number 154 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SemWare Announces QEdit Advanced the "quick editor"

QEdit is one of the smallest, fastest, multi-file editors available for IBM compatible computers and has many features to enhance your productivity.

- Lightning fast, especially in file loading and saving, screen display, cursor movement, and global searching and replacing.
- Edit multiple files, as many and as large as available memory allows. QEdit can use all available memory, up to 640KB. Ideal for multi-file languages and all those .PRG files.
- Windowing allows up to 7 files or 7 views of the same file to be seen at once.
- You do not have to "learn" another editor with QEdit. QEdit's commands can be configured to any keys you like.
- Screen colors, tab settings, default file type and many other options can be configured the way YOU like it.
- Up to 99 named scratch buffers may be created and edited. Commands are available for easy cut and paste operations.
- Invoke a DOS "shell" or execute DOS commands directly from QEdit.
- Both line and stream block marking are supported.
- Keyboard macros allow automation of commonly used command sequences or text entry.
- Commands to shift an entire block left or right, to adjust indentation levels when using structured languages.
- Garbage stack provides restoration of the last "n" (user-settable) deletions.

Introductory Price
49.95
Mastercharge & Visa Accepted
Add $3.00 Shipping
Georgia Residents please include 5% sales tax

QEdit Advanced requires an IBM PC/XT/AT or Compatible, IBM PS/2 30 - 80, 128KB of available memory. Please specify disk format required.

To Order: Heritage Papers, Danielsville, GA 30633
Phone Orders: (404) 789-3226

Number 148 on the Reader Service Card

NOW AVAILABLE IN C TOO!

Write A Data Base Program
(end to end)
in 10 minutes in TPascal 4.0
using
TURBO Programmer
(formerly Turbo GhostWriter)

Perfect for creating complex business applications!
Ideal for BASIC programmers who find TPascal too tough!
Spec your customer in the morning - Show a demo in the afternoon!

Features
- screen editor and painter
- automatic programmer documentation
- automatic data checking/validation
- plenty of "hooks" for customizing
- unlimited technical support

More features
- automatic B-tree indexing
- consistent user interface
- automatic context-sensitive help
- relational model to show customizing
- 30 day money-back guarantee (less $14 shipping/handling)

No questions to answer. Just draw your screens the way you want them to appear, tell Turbo Programmer how to set up the indexes and that's it... Running 4.0 code in 6 seconds with no programming. Regular price $450.

Now Only $289 Orders & Info 800 227-7681

ASCII 3239 Mill Run, Raleigh, NC 27612-4135

Number 170 on the Reader Service Card

number, (3) the starting date and time, and (4) the *per diem* and travel pay (if any).

Update your resume and mail it out to all the shops. Then, at the end of your assignment, you need only make a few calls to start the search for your next assignment.

Keep a detailed log of what you do. Some agencies require that you submit a description of your accomplishments during the pay period along with your hours. Typically they expect a paragraph describing what you did for each day of the week. Some expect you to submit only your hours. In any case, keep a log for yourself. At the end of each day write down what you did. Don't try to reconstruct your week at the end of the week.

What's Hot

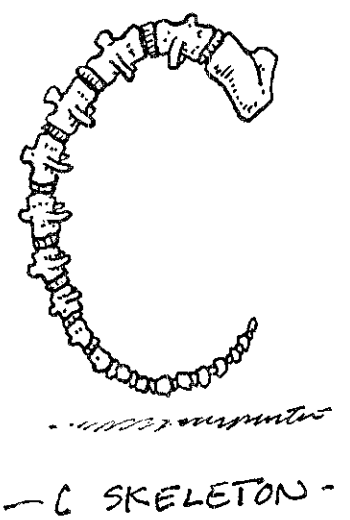
In my experience what's hot here in the Northeast is C/Unix, Unix internals, MS Windows, X Windows, LANs (Local Area Networks), PL/1, and database, particularly dBase, with C.

Conclusion

Don't quit your job to become a job-shopper. If what I've described appeals to you, wait until a time when you would naturally separate from your employer—then give job-shopping a try. Consider using job-shopping as a way of trying several companies before selecting one to settle down with.

If you are interested in working on your own, but aren't ready to be a consultant, job-shopping may be right for you. ♦

Oleg Panczenko, a software engineer, got his first programming job in 1979 after submitting a hand-written resume to his prospective employer. He has modeled ink jets, maintained a semiconductor-laser parameter database, programmed for a coordinate measuring-machine manufacturer, and worked on the user interface of a database product. Oleg spent 1987 job-shopping, then returned to direct work.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C CODE FOR THE PC

source code, of course

	Bluestreak Plus Communications (two ports, programmer's interface, terminal emulation)	$400
	CQL Query System (SQL retrievals plus windows)	$325
	GraphiC 4.1 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$325
	Barcode Generator (specify Code 39 (alphanumeric), Interleaved 2 of 5 (numeric), or UPC)	$300
NEW!	Vmem/C (virtual memory manager; least-recently used pager; dynamic expansion of swap file)	$250
	Aspen Software PC Curses (System V compatible, extensive documentation)	$250
	Greenleaf Data Windows (windows, menus, data entry, interactive form design)	$250
	PforCe++ (COM, database, file, user interface, & CRT C++ classes among others)	$345
	Vitamin C (MacWindows)	$200
NEW!	TurboTEX (TRIP certified; HP, PS, dot drivers; CM fonts; LaTEX)	$170
	Essential resident C (TSRify C programs, DOS shared libraries)	$165
	Essential C Utility Library (400 useful C functions)	$160
	Essential Communications Library (C functions for RS-232-based communication systems)	$160
	Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF)	$150
	Greenleaf Functions (296 useful C functions, all DOS services)	$150
	OS/88 (U**x-like operating system, many tools, cross-development from MS-DOS)	$150
	ME Version 2.0 (programmer's editor with C-like macro language by Magma Software; Version 1.31 still $75)	$140
	Turbo G Graphics Library (all popular adapters, hidden line removal)	$135
	PC Curses Package (full Berkeley 4.3, menu and data entry examples)	$120
	CBTree (B+tree ISAM driver, multiple variable-length keys)	$115
	Minix Operating System (U**x-like operating system, includes manual)	$105
	PC/IP (CMU/MIT TCP/IP implementation for PCs)	$100
	B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
	The Profiler (program execution profile tool)	$100
	Entelekon C Function Library (screen, graphics, keyboard, string, printer, etc.)	$100
	Entelekon Power Windows (menus, overlays, messages, alarms, file handling, etc.)	$100
NEW!	QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
	Wendin Operating System Construction Kit or PCNX, PCVMS O/S Shells	$80
	C Windows Toolkit (pop-up, pull-down, spreadsheet, CGA/EGA/Hercules)	$80
	Professional C Windows (windows and keyboard functions)	$80
	JATE Async Terminal Emulator (includes file transfer and menu subsystem)	$80
	MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	$80
	WKS Library (C program interface to Lotus 1-2-3 program & files)	$80
	Professional C Windows (lean & mean window and keyboard handler)	$70
	Quincy (interactive C interpreter)	$60
	EZ_ASM (assembly language macros bridging C and MASM)	$60
	PTree (parse tree management)	$60
	HELP! (pop-up help system builder)	$50
	Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card)	$50
	Make (macros, all languages, built-in rules)	$50
	Vector-to-Raster Conversion (stroke letters & Tektronix 4010 codes to bitmaps)	$50
	Coder's Prolog (inference engine for use with C programs)	$45
	C-Notes (pop-up help for C programmers ... add your own notes)	$40
	Biggerstaff's System Tools (multi-tasking window manager kit)	$40
	PC-XINU (Comer's XINU operating system for PC)	$35
	CLIPS (rule-based expert system generator, Version 4.1)	$35
	TELE Kernel or TELE Windows (Ken Berry's multi-tasking kernel & window package)	$30
	Clisp (Lisp interpreter with extensive internals documentation)	$30
	Translate Rules to C (YACC-like function generator for rule-based systems)	$30
	6-Pack of Editors (six public domain editors for use, study & hacking)	$30
	Crunch Pack (a dozen file compression & expansion programs)	$30
	ICON (string and list processing language, Version 7)	$25
NEW!	FLEX (fast lexical analyzer generator; new, improved LEX)	$25
	LEX (lexical analyzer generator; an oldie but a goodie)	$25
	Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	$25
	AutoTrace (program tracer and memory trasher catcher)	$25
	C Compiler Torture Test (checks a C compiler against K & R)	$20
	Benchmark Package (C compiler, PC hardware, and Unix system)	$20
	TN3270 (remote login to IBM VM/CMS as a 3270 terminal on a 3274 controller)	$20
	A68 (68000 cross-assembler)	$20
	List-Pac (C functions for lists, stacks, and queues)	$20
	XLT Macro Processor (general purpose text translator)	$20

Data

	WordCruncher (text retrieval & document analysis program)	$275
	DNA Sequences (GenBank 52.0 including fast similarity search program)	$150
	Protein Sequences (5,415 sequences, 1,302,966 residuals, with similarity search program)	$60
	Webster's Second Dictionary (234,932 words)	$60
	U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	$35
	The World Digitized (100,000 longitude/latitude of world country boundaries)	$30
	KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format)	$30
	USNO Floppy Almanac (high-precision moon, sun, planet & star positions)	$20
	NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	$15
	U. S. Map (15,701 points of state boundaries)	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409 USA

acw!info@uunet.uu.net

Voice: (512) 258-0785
BBS: (512) 258-8831
FidoNet: 1:382/12

Free shipping on prepaid orders **For delivery in Texas add 7%** **MasterCard/VISA**

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Windowing Systems Overview

by
Frank D. Greco

Nowadays it seems that most commercial end-user applications running on a DOS-OS/2 powered PC or Unix workstation use a formal window system—Microsoft Windows or X Windows—or at the very least some sort of pop-up/pull-down/pull-across menu-based user interface. As we move into the momentous era of multitasking and multi-MIP microcomputers with huge monitors, a window system model would seem to be mandatory.

Unfortunately, a standard window system still eludes us. From the personal computer world we have products such as Microsoft's Windows, its OS/2 manifestation the Presentation Manager, Digital Research's GEM, and Apple's now-ubiquitous Macintosh interface—each with its own array of installed customers and applications. On the Unix side, we have very sophisticated network-based systems such as MIT's X Windows, Silicon Graphics' 4Sight, Carnegie-Mellon's Andrew, and Sun Microsystems' NeWS. Right now, none of these disparate window systems can be considered truly standard.

Window Pains

None of these systems provides a clear or clean answer to the user interface question. Microsoft's Windows/Presentation Manager has been criticized for its unwieldiness (450 library calls). It appears to be the product of committee design and occasionally lacks an intuitive feel. I know of several large companies who just gave up on MS Windows because of its complexity.

The Apple Macintosh interface is much admired, but it is not truly multitasking and is proprietary to say the least.

While it's very easy to use, GEM lacks the backing of a major software house.

X Windows requires a multitasking operating system such as Unix, and it can't handle kerned fonts—a major flaw. Until Adobe Systems' Display PostScript is incorporated into X Windows, WYSIWYG products that use kerned outline fonts can't be built. How Display PostScript will be incorporated into X Windows has yet to be determined. And I have yet to see an X Windows application that equals the visual aesthetics of a Macintosh application. Any window system that hopes to be universally accepted must *at least* look as good as the Macintosh interface.

Programming for Sun's NeWS requires knowledge of PostScript. And while NeWS has the technical potential for success, it does not yet have a rich programmer toolkit, a requirement for any successful user interface.

The Origins of Windows

While the popularity of using windows can plausibly be attributed to the Apple Macintosh, Dan Ingalls came up with the original concept of splitting up the physical screen into separate areas called windows at Xerox's Palo Alto Research Center (PARC). This was in the early 1970s, and windows were developed for use in the Smalltalk environment. The invention of the mouse by Doug Englebart (who also invented the word processor) in the 1960s also contributed to the popularity of a windows interface. And I.E. Sutherland developed the direct man-machine graphical interface with his Sketchpad, a prototypical MacPaint/MacDraw system circa 1963. It is the combination of these three major innovations that has made window systems so successful.

When Is a Window Not a Window?

It should be noted (and has been noted in my Redirections column before) that the term "window" is actually used somewhat incorrectly in the popular computer literature. The term "window" in the standard computer graphics sense refers to a rectangular area of interest in the real-world coordinate space that the programmer/user wants to display. A section of screen real estate that maps this area is called a "viewport," but in everyday computer parlance, the term "window," rather than "viewport," has become more associated with a section of screen. This unfortunate association gets confusing and cumbersome when graphics libraries (such as PHIGS+ or GKS) are implemented over a window system. Recognizing this ambiguity, Sun Microsystems has come up with yet another word for a segment of the screen—Sun calls a segment of the screen a "canvas" (yet they don't talk about "canvas management" in their documentation!).

The Architecture of a Window System

From my perspective, a window system is composed of four basic layers:

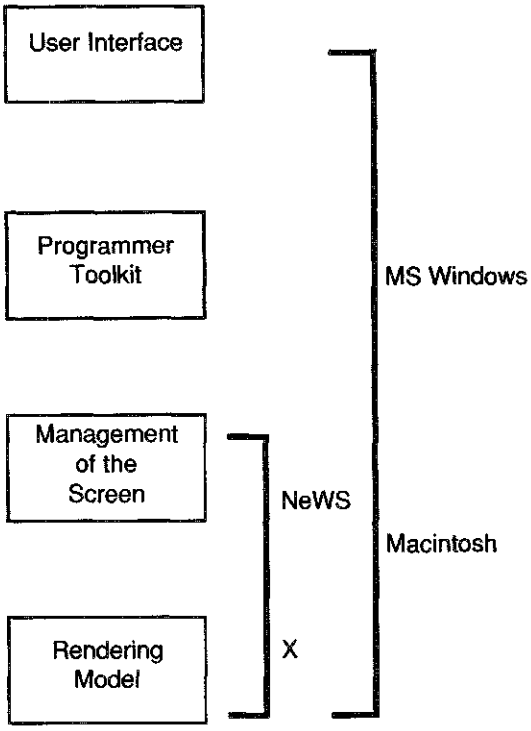
- User interface/application layer
- Programmer toolkit
- Management and synchronization of screen real estate
- Image rendering model

The user interface layer defines the application's so-called "look and feel." How scroll bars react to mouse events, how a window is made iconic, methods of resizing a window, etc., are all components of this abstraction level. The programmer toolkit is the collection of software primitives used to implement any particular user interface. The next lower level is the screen management layer—the heart of the window system. The management layer is responsible for interpreting incoming information from the toolkit layer (through either the stack or a more formal protocol) and for mediating access among processes or pseudo-processes to the physical screen. The rendering model is the layer that finally puts graphical images, color or monochrome, onto a physical screen.

Most window systems that currently run with the help of the Unix operating system have layers defined that clearly. For example, Sun Microsystems' NeWS rendering model is PostScript. Its management layer is called the NeWS server. One possible programmer toolkit for NeWS is called the NeWS Development Environment (NDE). One possible user interface that you can build using NDE is called "Open Look." The X Windows system uses similar design layers.

The architecture of PC-originated window systems, on the other hand, is not so clearly stratified. Besides being a base window management system, Microsoft Windows already predefines the user interface and interface toolkit. So does Apple's Macintosh. This forces the programmer to use only a single "look and feel" application interface. Such a restriction can be an asset if the user interface is a good one; witness the tremendous success of the Mac interface (especially with non-computer types). Digital Research's GEM doesn't define a user interface layer at all. Using the same GEM

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



programmer toolkit available from Digital Research, a developer can create quite different "look and feel" interfaces.

Not having clearly separate layers of responsibilities makes it difficult to wrench the programmer toolkit from either MS Windows or the Mac and port it to another windowing environment. Microsoft CEO Bill Gates is making a pitch to persuade the Unix-based X/Open organization to use the Presentation Manager as the standard Common Application Environment for Unix applications. I doubt he will succeed. Besides Unix-related political problems with IBM and concerns regarding putting the Presentation Manager specification into the public domain, *cleanly* separating the Presentation Manager programmer toolkit from the rest of the window system won't be easy. Notice that Apple is not exactly having an easy time porting the Macintosh interface over to A/UX (rumor has it that most of this work is based on the exclusively licensed software from the now-defunct Cadmus corporation).

A Very Sybil Window System

In addition, MS Windows also attempts to be a crutch for MS-DOS in its quest to be a real operating system (by definition, an operating system must have process management, which MS-DOS does not have). MS Windows manages memory, does (cooperative) multitasking, mediates resources, performs message passing interprocess communication, revectors INT 21H and many BIOS entry points—in short, it's trying to be the operating system that MS-DOS isn't. Not only are its architectural responsibilities confusing, but Microsoft also can't make up its mind whether the system should be called Presentation Manager or Windows. "Well if it uses OS/2, call it Presentation Manager; otherwise, call it Windows . . . Windows is virtually identical to PM." Heck, just call it one thing or the other!

You've Got the "Look"

When you hear that NeWS and X Windows will provide a standard look to all Unix applications, understand that that's not entirely correct. By definition, both NeWS and X Windows are window systems, not user interfaces. Recently, AT&T and Sun have made news by jointly announcing the availability of the "Open Look" Xerox PARC-style (bet you thought I was gonna say "Mac-style") user interface specification. Sun developed the Open Look specification with licensed help from Xerox (and has implemented a prototypical version), and AT&T will license implementations of the specification as commercial products. While not revolutionary, the Open Look specification represents a consistent "look and feel" for Unix application software. In fact, the specification of Open Look does not require any particular window system or operating system as the underlying engine and can be implemented even in an OS/2 environment. According to AT&T, first drafts of the specification will circulate among developers in July.

Work is now underway to develop programmer toolkits to assist in building Open Look. The X Windows toolkit, called "Xt," is available on the core distribution tape from MIT. Other higher level toolkits for X Windows, such as the Andrew toolkit from Carnegie Mellon University, Xrlib from Hewlett-Packard, and Interviews from Stanford University, are available on the user contributed tape from MIT. Each tape costs $200 and includes all source code. The NeWS toolkit, NDE, is an object-oriented toolkit in the Smalltalk tradition currently under development at Sun Microsystems. Because of a large installed base of application programs, Sun is also working on an enhanced toolkit for their older and archaic SunView proprietary window system so that it can support Open Look.

The Central Part of a Window System

The heart of a window system lies in its mechanism for interpreting communication between and within active processes and mediating access to the physical screen.

Microsoft Windows does this through its GDI (Graphics Device Interface) and USER modules. The third module in MS Windows, the KERNEL module, basically handles process (co-routine) management and memory management. MS Windows function calls are Pascal-flavored functions that call one of the three basic modules, which invokes either a Windows device driver or an operating system function. The communication "protocol" used by MS Windows is simply the standard use of the stack. I'm not yet convinced that the choice of a Pascal calling sequence for MS Windows functions is a good idea.

Gotta Network Yourself These Days

As opposed to the Macintosh and MS Windows systems, both X Windows and NeWS are server-based, networkable window systems. This does not mean that programs that obtain information over a network can't be built with MS Windows or the Macintosh. Quite the contrary—some very innovative network-based programs can be built with the Macintosh and MS Windows. A network window system just implies that networking applications can be built much more easily because a network is a natural constituent of the window system.

A (redirectable) network window server is logically "connected" to a workstation's screen and communicates with application clients anywhere on the network. Unlike traditional servers, as in a file server, window servers exist in every workstation that requires the rendering of graphical images. X Windows clients talk to the X server via a numerical high-level protocol (the X Protocol) that is generated through the X library. The NeWS protocol is an interesting one. NeWS clients talk to the NeWS server by sending PostScript program segments to it. In effect, you are downloading a program when you communicate with a NeWS server. The use of PostScript gives you the advantage of an effective communications

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Unbelievable!

SOURCER™

■ SEE HOW PROGRAMS WORK
■ EASILY MODIFY PROGRAMS

SOURCER™ creates detailed commented source code and listings from memory and executable files. Built in data analyzer and simulator resolves data across multiple segments and provides detailed comments on interrupts and subfunctions, I/O ports and much more. Determines necessary assembler directives for reassembly. Complete support for 8088 through 80286, V20/V30, 8087, and 80287 instruction sets. We welcome comparisons with any other product, because no product comes close to the ease of use and output clarity of SOURCER.

On my list of programs that I simply won't do without!
—Robert Hummel, Senior Technical Editor, PC Magazine

SAMPLE OUTPUT

Fully automatic Program header

Assembler directives

Determines data areas and type

Detailed comments

Simulator follows segment changes

Easy to read format

```
resetprn.lst    ResetPRN v1.01                Sourcer Listing    19-Apr-88   4:05 pm      Page 1

                                PAGE  60,132
                                ;-----------------------------------------------------------------
                                ;                          RESETPRN
                                ;
                                ;       Created:   15-Apr-88
                                ;       Version:   1.01
                                ;-----------------------------------------------------------------

= 0008                          data_1e         equ     8               ; (0040:0008=378h)

                                ;---------------------------------------------------- seg_a -----
                                seg_a           segment para public
                                                assume cs:seg_a, ds:seg_a, ss:stack_seg_b

                                resetprn        proc    far
                                start:
658E:0000                       ;
658E:0000  EB 23                                jmp     short loc_1
658E:0002  52 65 73 65 74 50                    db      'ResetPRN v1.01', 0Dh
658E:0008  52 4E 20 76 31 2E
658E:000E  30 31 0D
658E:0011  0040                 data_2          dw      40h
658E:0013  0D 0A 52 65 73 65    data_3          db      0Dh, 0Ah, 'Reset Printer? $'
658E:0019  74 20 50 72 69 6E
658E:001F  74 65 72 3F 20 24
658E:0025                       loc_1:
658E:0025  0E                                   push    cs
658E:0026  1F                                   pop     ds
658E:0027  BA 0013                              mov     dx,offset data_3  ; (658E:0013=0Dh)
658E:002A  B4 09                                mov     ah,9
658E:002C  CD 21                                int     21h             ; DOS Services  ah=function 09h
                                                                        ;  display char string at ds:dx
658E:002E  B4 01                                mov     ah,1
658E:0030  CD 21                                int     21h             ; DOS Services  ah=function 01h
                                                                        ;  get keybd char al, with echo
658E:0032  3C 79                                cmp     al,79h          ; 'y'
658E:0034  75 16                                jne     loc_3           ; Jump if not equal
658E:0036  8E 1E 0011                           mov     ds,data_2       ; (658E:0011=40h)
658E:003A  8B 16 0008                           mov     dx,ds:data_1e   ; (0040:0008=378h)
658E:003E  83 C2 02                             add     dx,2
658E:0041  B0 08                                mov     al,8
658E:0043  EE                                   out     dx,al           ; port 37Ah, printer-2 control
                                                                        ;  al = 8, initialize printer
658E:0044  B9 8000                              mov     cx,8000h
658E:0047                       locloop_2:
658E:0047  E2 FE                                loop    locloop_2       ; Loop if cx > 0
658E:0049  B0 0C                                mov     al,0Ch
658E:004B  EE                                   out     dx,al           ; port 37Ah, printer-2 control
                                                                        ;  al = 0Ch, init & strobe off
658E:004C                       loc_3:
658E:004C  B4 4C                                mov     ah,4Ch          ; 'L'
658E:004E  CD 21                                int     21h             ; DOS Services  ah=function 4Ch
                                                                        ;  terminate with al=return code
                                resetprn        endp
                                seg_a           ends

                                ;---------------------------------------------------- stack_seg_b ----
                                stack_seg_b     segment para stack
6593:0000  00C0[ff]                             db      192 dup (0FFh)
                                stack_seg_b     ends

                                                end     start
```

(Source code output and inline cross reference can also be selected)

BIOS SOURCE

■ CHANGE AND ADD FEATURES
■ CLARIFY INTERFACES

for PS/2, AT, XT, PC, and Clones

The BIOS Pre-Processor™ with SOURCER provides the first means to obtain accurate legal source listings for any BIOS! Identifies entry points with full explanations. Resolves PS/2's multiple jumps for improved clarity. Provides highly descriptive labels such as "video_mode" and much more. Fully automatic.

SOURCER $99.95 BIOS Pre-Processor* $49.95 SOURCER w/BIOS Pre-Processor $139.95

(Outside USA, Add $15 Shipping & Handling; CA Residents add local sales tax 6, 6 5 or 7%; *requires SOURCER)

All our products come with a 30 day money back satisfaction guarantee. Not copy protected. To order or receive additional information just call!

800-538-8157 x811 (outside Calif.)
800-672-3470 x811 (inside Calif.)

V COMMUNICATIONS
3031 Tisch Way, Suite 200, Dept. J2, San Jose, CA 95128 (408) 296-4224

PS 2, AT, XT, and PC are trademarks of IBM Corp.

Number 125 on the Reader Service Card

bandwidth increase (once something is downloaded into the server), natural data compression, and tremendous extensibility.

X Windows versus NeWS

While the notion of a window manager is inseparable from the design of MS Windows, both X Windows and NeWS are structured so that window managers are seen as just another window system client application. Several types of window managers can be written for either X or NeWS. This separation of window placement policy from management of the screen real estate is an interesting idea. Such a distinction means that different types of window managers—tiled or overlapping, for instance—can be implemented as user programs that communicate with the window system server. X Windows, however, assumes that windows can be rectangular only; NeWS doesn't make that assumption. A window in X Windows also requires the use of an expensive Unix heavyweight process. Since NeWS uses lightweight processes (unfortunately, nonpreemptive, as in MS Windows), the creation of a window in NeWS is very fast and cheaper in terms of resources. X Windows requires a multitasking operating system. NeWS is implemented as one heavyweight process and so can be implemented on a single-tasking operating system. In fact, besides running on Unix machines, NeWS is currently running on the Atari and the Mac.

Windows of Opportunity

I've only touched the surface of what window systems are and how they work. It's very clear that window systems and graphical user interfaces will be the norm on most computer architectures in the near term future (that was clear years ago). Given that we will shortly have 100 MIP workstations on the desktop, who knows what the next viable user interface will be. Ain't this fun?

Before I Go, a Soapbox to Remember Me by

The Macintosh's user interface has been the standard by which every other interface is judged, and this is unfortunate in certain respects. If we aspire merely to the familiar, the level of user interface sophistication will remain frozen at the Macintosh level. While the Macintosh interface is currently one of the most intuitive graphical user interfaces for computers, it surely is not the last word in man-machine interaction. ♦

Frank D. Greco is an independent consultant who does industry analysis, systems consulting, and technological forecasting for Fortune 500 companies. Readers can contact Frank at the following address:

Frank D. Greco
c/o Crossroads Technologies, Inc.
P.O. Box 530
Fanwood, NJ 07023
(201) 754-7820

PRODUCED 2024 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

KEEP UP WITH THE OS/'s

Megabytes of Memory and 32-Bit Performance for DOS.

If you thought the only way to protected mode was the big move to OS/2...We have good news! You can gain the benefits of protected mode the easy way with OS/286™ and OS/386™. These tools for C, Fortran, Pascal and assembly language programmers permit rapid conversion of existing DOS applications from "real" 8086 mode to "protected" 286 and 386 mode. They don't replace or modify DOS, but extend it to protected mode. This way you get multi-megabytes of directly addressable memory (16Mb-286, 4Gb-386) with the compiler, TSRs, device drivers, graphic routines, etc. you use today.

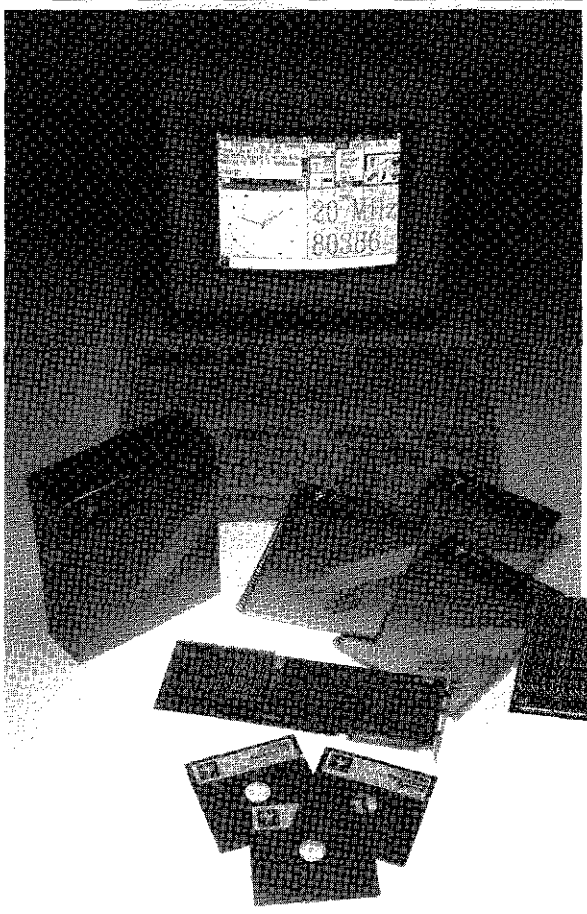
OS/286 and OS/386 are the only DOS extenders that span both the 286 and 386 processors. They run on the widest array of AT and 386 machines, with 32-bit capability *today* on 386s that yields twice the performance of 16-bit mode.

OS/286 and OS/386 are the preferred solutions for developers of high performance memory-intensive applications, including CADKEY, CASE, and Gold Hill, and premier language developers Lahey and MetaWare.

Our optional TOUCHDOWN™ BIOS supplement provides fast and reliable protected mode operation on *any* 286 system, even those with problems resetting the 286. (Ever notice how few existing machines Operating System/2 runs on?) TOUCHDOWN is not required for most major brand AT clones, but for the older machines it is a lifesaver!

If your applications are running out of memory or need more speed, enhance them now without abandoning your investment in DOS.

Special
$49.50
Evaluation Offer



Check out for yourself the benefits of protected mode. Our $49.50 "sampler special" is a complete OS/286 Developer's Kit, but with a time limited, non-distributable kernel. There's no better way to learn about the outstanding features of OS/286 and OS/386 than to try them. Of course, the $49.50 is applicable to the purchase of the full OS/286 (16-bit) or OS/386 (32-bit) kit at $495 for either one.

The OS/x86 Developer's Kits include support for popular, C, Fortran, Pascal, Lisp, Prolog, compilers, and assemblers from: MetaWare, Lahey, Microsoft, Lattice, Gold Hill, LogicWare and Phar Lap. A number of other packages are also supported including PLINK86, Halo & GSS Graphics and DESQview.

The HummingBoard® turns any XT, AT or 386 into the fastest, most capable 386 system available. The dual processor architecture provides up to 900K each for multiple real mode applications (Lotus, etc.) which can co-reside with protected mode applications on the HummingBoard's 20MHz 386, while *debuggers, editors,* networks run concurrently on the base processor. Applications using OS/x86 automatically split tasks between the Hummingboard and the base. This can provide a 5-10x speed-up with a Hummingboard in an AT and up to a 2x performance boost with it in a fast 386 system, such as the Compaq 386/20. Hummingboards available with up to 24Mb, 80387.

Prices start at just $1,495

A.I. Architects, Inc.
One Kendall Square, Cambridge, MA 02139
TEL (617) 577-8052 FAX (617) 577-9774
Credit card orders only – 24 hours. 617-577-1305

HummingBoard® is a registered trademark and OS/286, OS/386 and TOUCHDOWN are trademarks of A.I. Architects, Inc., PLINK86 is a trademark of Phoenix Corp., HALO is a registered trademark of Media Cybernetics, Inc., DESQview is a trademark of Quarterdeck Office Systems. Compaq Deskpro 386 is a trademark of Compaq Computer Corp.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Reading VGA Memory

by
Michael Abrash

Installment 7: In which we learn to read VGA memory and encounter the strange but useful Color Compare.

Well, it's taken six installments of On Graphics, but we've finally covered all four write modes of the VGA. Now it's time to tackle the VGA's two read modes. While the read modes aren't as complex as the write modes, they're nothing to sneeze at. In particular, read mode 1 (also known as color compare mode) is rather unusual and not particularly intuitive.

You may well ask, isn't *anything* about programming the VGA straightforward? Well . . . no. But then, clearing up the mysteries of VGA programming is what these articles are all about, so let's get started.

Read Mode 0

Read mode 0 is actually relatively uncomplicated, given that you understand the four-plane nature of the VGA. (If you don't understand the four-plane nature of the VGA, get your hands on *PJ* issues 5.1–5.4 and 6.1–6.2.) Read mode 0, the read mode counterpart of write mode 0, lets you read from one (and only one) plane of VGA memory at any one time.

Read mode 0 is selected by setting bit 3 of the Graphics Mode register (Graphics Controller register 5) to 0. When read mode 0 is active, the plane that supplies the data when the CPU reads VGA memory is the plane selected by bits 1 and 0 of the Read Map register (Graphics Controller register 4). When the Read Map register is set to 0, CPU reads come from plane 0 (the plane that normally contains blue pixel data). When the Read Map register is set to 1, CPU reads come from plane 1; when the Read Map register is 2, CPU reads come from plane 2; and when the Read Map register is 3, CPU reads come from plane 3.

That all seems simple enough; in read mode 0, the Read Map register acts as a selector among the four planes, determining which one of the planes will supply the value returned to the CPU. There is a slight complication, however, in that the value written to the Read Map register in order to read from a given plane is not the same as the value written to the Map Mask register (Sequence Controller register 2) in order to write to that plane.

Why is that? Well, in read mode 0, one and only one plane can be read at a time, so there are only four possible settings of the Read Map register: 0, 1, 2, or 3, to select reads from plane 0, 1, 2, or 3. In write mode 0, by contrast (in fact, in any write mode), any or all planes may be written to at once since the byte written by the CPU can "fan out" to multiple planes. Consequently, there are not four but sixteen possible settings of the Map Mask register. The setting of the Map Mask register to write only to plane 0 is 1; to write only to plane 1 is 2; to write only to plane 2 is 4; and to write only to plane 3 is 8.

As you can see, the settings of the Read Map and Map Mask registers for accessing a given plane don't match. The code in Listing 1 illustrates this. Listing 1 simply copies a sixteen-color image from system memory to VGA memory, one plane at a time, then animates by repeatedly copying the image back to system memory, again one plane at a time, clearing the old image and copying the image to a new location in VGA memory. Note the differing settings of the Read Map and Map Mask registers.

By the way, the code in Listing 1 is intended only to illustrate read mode 0,

and is, in general, a poor way to perform animation since the method is slow and tends to produce flicker. In a future issue of *PJ*, I'll take a look at VGA animation techniques.

As you might expect, neither the read mode nor the setting of the Read Map register affects CPU *writes* to VGA memory in any way.

An important point regarding reading VGA memory involves the VGA's latches. (Remember that each of the four latches stores a byte for one plane; on CPU writes, the latches can provide some or all of the data written to display memory, allowing fast scrolling and efficient pixel masking.) Whenever the CPU reads a given address in VGA memory, each of the four latches is loaded with the contents of the byte at that address in its respective plane. Even though the CPU receives data from only one plane in read mode 0, all four planes are always read and the values read are stored in the latches. This is true in read mode 1 as well. In short, whenever the CPU reads VGA memory in any read mode, all four planes are read and all four latches are loaded.

Finally, before we go on to read mode 1, I'd like to note that the EGA

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

supports slightly different versions of read mode 0 and read mode 1. The EGA's read modes are compatible with the VGA's read modes so long as you don't set bit 2 of the EGA's Read Map register to 1, and so long as you don't touch the EGA's Graphics Position registers (at ports 3CCh and 3CAh).

Read Mode 1

Read mode 0 is the workhorse read mode, but it's got an annoying limitation: Whenever you want to determine the color of a given pixel in read mode 0, you have to perform four VGA memory reads, one for each plane, and then interpret the four bytes you've read as eight 16-color pixels. That's a lot of programming; the code is also likely to run slowly, all the more so because the VGA takes an average of 1.1 microseconds to complete each memory read and read mode 0 requires four reads in order to read the four planes. (1.1 microseconds may not sound like much, but on an 8-MHz AT, it's 9 clock cycles, and on a 20-MHz Compaq 80386 machine, it's 22 clock cycles!)

Read mode 1, or color compare mode, provides special hardware assistance for determining whether a pixel is a given color. With a single read mode 1 read, you can determine whether up to eight pixels are a specific color, and you can even specify any or all planes as "don't care" planes in the pixel color comparison.

Read mode 1 is selected by setting bit 3 of the Graphics Mode register (Graphics Controller register 5) to 1. In its simplest form, read mode 1 compares the cross-plane value of each of the eight pixels at a given address to the color value in bits 3–0 of the Color Compare register (Graphics Controller register 2) and returns a 1 to the CPU in the bit position of each pixel that matches the color in the Color Compare register and a 0 for each pixel that does not match.

That's certainly interesting, but what's read mode 1 good for? One obvious application is in implementing flood-fill algorithms since read mode 1 makes it easy to tell when a given byte contains a pixel of a boundary color. Another application is in detecting on-screen object collisions, as illustrated by the code in Listing 2.

Still and all, there aren't all that many uses for basic color compare operation. There is, however, a genuinely odd application of read mode 1 that's worth knowing about; in order to understand that, we must first look at the "don't care" aspect of color compare operation.

As described above, during read mode 1 reads, the color stored in the Color Compare register is compared to each of the eight pixels at a given address in VGA memory. But—and it's a big but—any plane for which the corresponding bit in the Color Don't Care register is a 0 is always considered a color compare match, regardless of the values of that plane's bits in the pixel and in the Color Compare register.

There is a genuinely odd application of read mode 1 that's worth knowing about.

Let's look at this another way. A given pixel is controlled by four bits, one in each plane. Normally (when the Color Don't Care register is 0FFh), the color in the Color Compare register is compared to the four bits of each pixel; bit 0 of the Color Compare register is compared to the plane 0 bit of each pixel, bit 1 of the Color Compare register is compared to the plane 1 bit of each pixel, and so on; that is, when the lower four bits of the Color Don't Care register are all set to 1, all four bits of a given pixel must match the Color Compare register in order for a read mode 1 read to return a 1 for that pixel to the CPU.

But if any bit of the Color Don't Care register is 0, the corresponding bit of each pixel is unconditionally considered to match the corresponding bit of the Color Compare register. You might think of the Color Don't Care register as selecting exactly which planes should matter in a given read mode 1 read. At the extreme, if all bits of the Color Don't Care register are 0, read mode 1 reads will always return 0FFh since all planes are considered to match all bits of all pixels.

Now, we're all prone to using tools the "right" way—that is, in the way in which they were intended to be used. By that token, the Color Don't Care register is clearly intended to mask one or more planes out of a color comparison, and as such, has limited use. However, the Color Don't Care register becomes far more interesting in exactly the "extreme" case described above, where all planes become "don't care" planes.

Why? Well, as I've said, when all planes are "don't care" planes, read mode 1 reads always return 0FFh. Now, when you AND any value with 0FFh, the value remains unchanged, and that can be awfully handy when you're using the bit mask to modify selected pixels in VGA memory. Recall that you must always read VGA memory to load the latches before writing to VGA memory when you're using the bit mask. Traditionally, two separate instructions—a read followed by a write—are used to perform this task. The code in Listing 2 uses this approach. Suppose, though, that you've set the VGA to read mode 1, with the Color Don't Care register set to 0 (meaning all reads of VGA memory will return 0FFh). Under these circumstances, you can use a single AND instruction to both read and write VGA

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

memory since ANDing any value with 0FFh leaves that value unchanged.

Listing 3 illustrates an efficient use of write mode 3 in conjunction with read mode 1 and a Color Don't Care register setting of 0. The mask in AL is passed directly to the VGA's bit mask (that's how write mode 3 works—see *PJ* 6.1 for details). Because the VGA always returns 0FFh, the single AND instruction loads the latches, leaves AL unchanged, and writes the value in AL to the VGA, where the value is used to generate the bit mask. This is faster, more compact, and more register-efficient than using separate instructions to read and write.

By the way, I commented in *PJ* 6.1 that I hadn't come across any particularly noteworthy applications for write mode 3. That's true no longer—Listing 3 shows how write mode 3 can make for excellent dot- and line-drawing code.

I hope I've given you a good feel for what color compare mode is and what it might be used for. Color compare mode isn't particularly easy to understand, but it's not that complicated in actual operation, and it's certainly useful at times; take some time to study the sample code and perform a few experiments of your own, and you may well find some useful applications for color compare mode in your graphics code.

A final note: The Read Map register has no effect in read mode 1, and the Color Compare and Color Don't Care registers have no effect either in read mode 0 or when writing to VGA memory.

Using Local Labels in REPT Blocks

I'm going to round out this installment with some odds and ends that have accumulated while I've been covering read and write modes. First, a truly annoying MASM 5.0 bug (at least, I think it's a bug): How the heck can I use local variables in REPT blocks? On page 223 of the MASM 5.0 manual I find the statement, "Macro operators, symbols declared with the LOCAL directive, and the EXITM directive can be used in repeat blocks."

Well, sure, the LOCAL directive can be used in a REPT block, and symbols declared with LOCAL can be used in repeat blocks—however, those symbols can't be used more than once in a given repeat block, which kind of defeats the purpose of using local symbols. I've also tried embedding REPT blocks inside macros and declaring symbols local in the macro, both inside and outside the REPT block, but that didn't work either. Listing 4 shows some of the approaches I've tried without success.

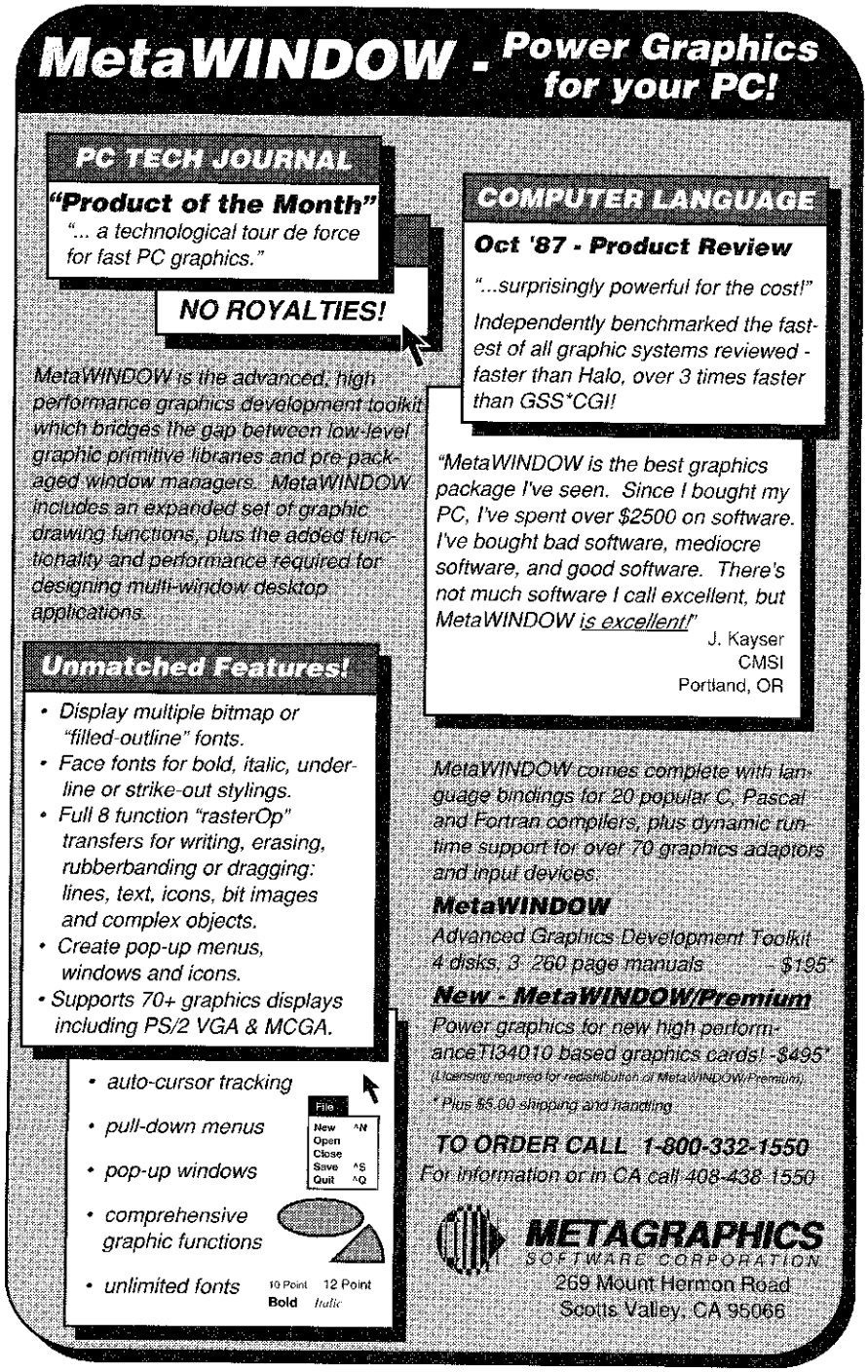
In the past, *PJ* readers have always come up with innovative solutions to the MASM problems I've encountered. Any help with this one would be appreciated, and would of course be shared with *PJ*'s readers.

Slow VGA Mode 13H Mode Sets

I've gotten reader feedback to the effect that mode 13h mode sets on IBM's VGA are extremely slow, to the point of interfering with code that performs frequent mode sets without clearing display memory (by way of setting bit 7 of AL high when invoking INT 10h, function 0). I haven't noticed this myself, but then I haven't done much work with mode 13h. Does anyone out there know more about this problem, and if so, do you have a workaround?

Reference Material

My primary reason for starting this On Graphics series was the then near-total absence of useful reference material on graphics adapters. That situation is now much eased by the appearance of Richard Wilton's excellent book, *Programmer's Guide to PC & PS/2 Video*


MetaWINDOW - Power Graphics for your PC!
PC TECH JOURNAL
"Product of the Month"
"... a technological tour de force for fast PC graphics."
NO ROYALTIES!
COMPUTER LANGUAGE
Oct '87 - Product Review
"...surprisingly powerful for the cost!"
Independently benchmarked the fastest of all graphic systems reviewed - faster than Halo, over 3 times faster than GSS*CGI!
MetaWINDOW is the advanced, high performance graphics development toolkit which bridges the gap between low-level graphic primitive libraries and pre-packaged window managers. MetaWINDOW includes an expanded set of graphic drawing functions, plus the added functionality and performance required for designing multi-window desktop applications.
"MetaWINDOW is the best graphics package I've seen. Since I bought my PC, I've spent over $2500 on software. I've bought bad software, mediocre software, and good software. There's not much software I call excellent, but MetaWINDOW is excellent!"
J. Kayser
CMSI
Portland, OR
Unmatched Features!
• Display multiple bitmap or "filled-outline" fonts.
• Face fonts for bold, italic, underline or strike-out stylings.
• Full 8 function "rasterOp" transfers for writing, erasing, rubberbanding or dragging: lines, text, icons, bit images and complex objects.
• Create pop-up menus, windows and icons.
• Supports 70+ graphics displays including PS/2 VGA & MCGA.
• auto-cursor tracking
• pull-down menus
• pop-up windows
• comprehensive graphic functions
• unlimited fonts
File
New ^N
Open
Close
Save ^S
Quit ^Q
10 Point 12 Point
Bold Italic
MetaWINDOW comes complete with language bindings for 20 popular C, Pascal and Fortran compilers, plus dynamic run-time support for over 70 graphics adaptors and input devices.
MetaWINDOW
Advanced Graphics Development Toolkit
4 disks, 3 260 page manuals $195*
New - MetaWINDOW/Premium
Power graphics for new high performance TI34010 based graphics cards! - $495*
(Licensing required for redistribution of MetaWINDOW/Premium)
* Plus $5.00 shipping and handling
TO ORDER CALL 1-800-332-1550
For information or in CA call 408-438-1550
METAGRAPHICS
SOFTWARE CORPORATION
269 Mount Hermon Road
Scotts Valley, CA 95066

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

It's Our Objective To Keep You Oriented

NEW

JOURNAL OF OBJECT-ORIENTED Programming

Editor: **Dr. Richard Wiener** *Univ. of Colorado*
Assoc. Editor: **Prof. David Thomas**, *Carleton Univ.*

An innovative bimonthly journal offering original and stimulating research in all areas of object-oriented programming.

The Journal of Object-Oriented Programming (JOOP) is designed to track and evaluate the advances in the object-oriented methodology.

Well-researched articles on topics pertinent to the field and your daily work will appear, such as:

- **problem-solving techniques**
- **reuseable OOP components**
- **latest language developments**
- **applications in artificial intelligence**
- **software maintenance**
- **unbiased product reviews**
- **up-to-the minute product news**

JOOP maintains a prestigious international editorial board from industry and academia, which include:

Adele Goldberg
ParcPlace Systems
Brad Cox
StepStone Corp.
Bjarne Stroustrup
AT & T Bell Labs.
Kurt Schmucker
Apple Computer

The journal's incisive articles are written by well-known academic researchers and premier software developers from around the world. Important upcoming articles include:

"Active Objects, An Access Oriented Framework for Object-Oriented Languages" Dr. Kasper Osterhye

"The Model View Controller Cookbook" Dr. Stephen Pope

"Using Objects to Package User Interface Functionality" Kurt Schmucker

"Designing Reusable Classes" Dr. Ralph Johnson
"Smalltalk-80 Based Computer Music Tools" Dr. Stephen Pope

"Communication, Scheduling and Resource Management in SINA—A New Object-Oriented Language" Dr. Armand Tripathi

JOOP is written for software developers, engineers, researchers and computer scientists. Get original research that's definitive, authoritative and applicable. Get objective coverage that's enlightening, reliable and functional.

Subscribe Today

Order toll free **1-800-345-8112** (7 days a week)
or mail in this coupon

Yes, send me JOOP at the following term

1 year (6 issues)
☐ individual - $49 ☐ institutional - $79
2 years (12 issues)
☐ individual - $89 ☐ institutional - $146

Order outside the U.S., add $24 a year for Air Service.

Check enclosed _____ Bill me _____
Charge my: Visa _____ Mastercard _____
Card # _____ Exp. Date _____
Signature _____

Your Name _____
Company _____
Address _____
City _____ State ____ Zip _____

Individual subscriptions must be paid with a personal check.
Make checks payable to **SIGS Publications**
Mail to: JOOP, 310 Madison Avenue, Suite 503, New York, New York 10017

Code 0588

Number **205** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Systems, from Microsoft Press. [See Miscellaneous, *PJ* 6.3.—Ed.] I haven't read the entire book (sorry about that, but I do after all already know much of the subject matter), but the chapters I have read are accurate and readable and come with large quantities of sample code. The book covers all current graphics standards, including CGA, EGA, VGA, and Hercules, and does a good job of it. Simply put, *Programmer's Guide to PC & PS/2 Video Systems* is far and away the best PC graphics reference I've seen to date.

There is one flaw with Wilton's book, however, although I very much doubt it's the author's doing: The listings are typeset in pale green and are virtually invisible without good lighting. It's genuinely a shame that so much useful sample code should be so hard to read.

Video Image Capture

David P. Holberton wrote with an interesting question about the image-capture capabilities of the EGA and VGA: Can the EGA or VGA easily be modified to allow video images to be captured directly into EGA/VGA display memory?

I surely didn't know the answer—hardware is not my field—but I asked several hardware engineers, and they were unanimous in their opinion that any such change would require major surgery to the EGA or VGA board. (They were also unanimous in their opinion that while *they* could make such a change, they didn't recommend it for other people.) The feature connectors at the top of the EGA and Display Adapter boards (which are not identical, by the way) are useless for this purpose since they provide no path to display memory. In short, the answer seems to be no. However, David's question was sparked by a paragraph from page 640 of Nelson Johnson's book *Advanced Graphics in C*, so perhaps there's something we're missing here. Possibly the modification involved the EGA's memory expansion connector; however, that would be of limited use since very few EGA clones have memory expansion connectors. In any case, contributions on this topic are welcome.

In Conclusion

Gee, we're actually done with the basics of accessing VGA memory! Fortunately, that still leaves a slew of interesting VGA topics, such as multiple font support, multiple text pages, 43- and 50-line text, smooth panning and scrolling, the split screen, the vertical interrupt, and color selection—not to mention actual applications, including my personal favorite, animation. Frankly, I don't know just what I'll cover next time—tune in and find out. See you then. ♦

Michael Abrash is a Senior Software Engineer for Orion Instruments, Inc., of Redwood City, California, where he is working on the design of advanced PC-based test instrumentation. Readers may write to Michael at

P.O. Box 39051
Mountain View, CA 94039

or drop him a line via MCI Mail (user name: MABRASH).

Code follows.

PLOT TEXT ON ANY GRAPHICS SCREEN!!

YES, we said **ANY** graphics screen, even VGA! ***Finally!*** **Xgraf** is a super set of low level assembly graphics routines that you call directly from Compiled Basic. The **Xgraf** kernel replaces Basic's confusing statements with consistent full featured calls specifically designed for the Basic programmer. **Xgraf** supports **Hercules, CGA, EGA, EEGA, VGA, and user defined screen sizes.** All modes can be accessed in real time - no recompiling!

Advanced features extend Basic graphic capabilities!

In addition to Basic's graphics features, **Xgraf's** ***advanced features*** include zooming, tiling, a virtual screen page, screen compression to disk, windowing, screen importing, *and much more!*

Finally! **Xgraf is only $99.00 + $4.00 s&h**
(Xgraf requires the Microsoft QuickBASIC compiler)

We *specialize* in libraries and tools for Compiled Basic. Our catalog features the ***Finally!*** family of programmer's libraries and *other top flight tools.*

KOMPUTERWERK - YOUR BASIC RESOURCE

KOMPUTERWERK
851 Parkview Blvd.
Pittsburgh, PA 15215

Call: 1-800-423-3400
PA & AK call (412) 782-0384
VISA and MC accepted!

Number **155** on the Reader Service Card

HGC TOOLS™

Hercules™
InColor Card
Graphics Card Plus
Graphics Card

MODE & REGISTER CONTROL: Routines provide flexible programming of the 6845 CRT controller chip used on the Hercules™ family of graphics cards, including ALL extension registers. Facilities efficiently control the various modes of operation of the cards in areas such as Text, RamFont,™ Graphics and Color processing.

COMPREHENSIVE HEADER FILES: Both C and Assembler "Include" files provided an extensive array of Register mnemonics, macro control routines and global data area control.

EMULATION LIBRARY: A library of graphics routines which extends the functionality of the Microsoft® QuickC™ / C5.x Graphics.LIB to the Hercules™ Cards, including full color processing on the InColor™ Card. A superset, the many functions provide additional features in support of the Hercules™ environment. The Microsoft® compiler(s) are required for use with this emulation library.

Soft Evolutions, Inc.
484 Lake Park Ave., #17
Oakland, CA 94610-2730
(415) 523-8247

INTRODUCTORY PRICE!
$34.95

Number **177** on the Reader Service Card

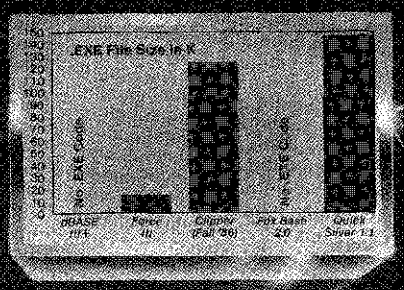
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

FORCE III

The Ultimate dBase Compiler

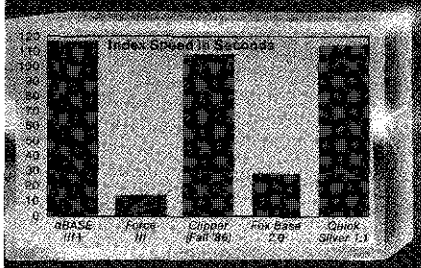
FORCE III is the world's fastest dBase compiler. It compiles and links the quickest, and the .EXE files it creates run so fast you'll swear you wrote your program in C. To give you an idea how fast, take a look at the *Index Speed* graph below. This graph compares indexing speeds of the different products using a database which contained 8,000 records. The database was indexed on the numeric field, NUM.

Best of all, FORCE III creates very small .EXE files. The .EXE files are *true* native code files and do not rely on any kind of memory resident library or interpreter! The reasons the .EXE files are so small are simple: FORCE III is an "intelligent" compiler and only uses those routines from the library that your program requires, and secondly, the entire library is written in Assembler, which makes the routines very small and very fast.



In the *Index Size* graph (below), the index files created by each product are compared. As you can see, not only is FORCE III the fastest at indexing, it also creates the smallest index files!

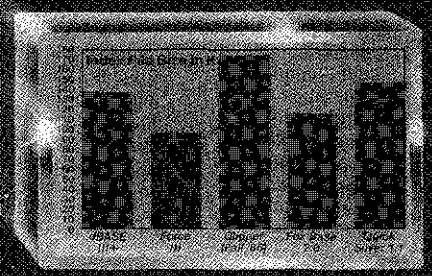
(Tests conducted on an IBM 8Mhz. AT with 20Mb hard disk.)



$129

Royalty Free and fully linkable!

In addition to being the fastest true dBase compiler on the market, FORCE III generated .EXE files carry no royalties whatsoever! And FORCE III compiled programs can be linked to Borland's Turbo C and other C compilers using the Intel/ Microsoft OBJ format. And linking to Assembler is a snap. We even included an ASCII graphics generator for easy screen and menu generation.



Arrays, UDFs, sound, new functions, and more!

Force III includes features you've come to expect in more expensive dBase compilers. For the programmer we have included I/O re-direction, FOR-NEXT loops, C like file primitives, and new data types. We've even included some Clipper compatible commands such as VALID.

Call now, 800-922-3001.

FORCE III comes with extras, like a 120 day warranty, an excellent reference manual and for the first 10,000 customers, we'll include the premier sourcework, **Advance dBASE III Plus Programming and Techniques** by Miriam Liskin (a $21.95 value). All this for just $129.00 (add $5.00 s&h). In Colorado call 303-444-1542. In Europe call London at 44-1-631-0548. Or write: Sophco, Inc., PO Box 7430, Boulder, Colorado, 80306-7430.

Products mentioned above are trademarked by their respective manufacturers.

Number 136 on the Reader Service Card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available on 360K, MS-DOS disk, $10.00 postpaid. ORDER BY PHONE — VISA/MC orders call (503) 747-0800. Mail order information on page 76.

Code from "Reading VGA Memory" by Michael Abrash

1

```
; *** Listing 1 ***
;
; Program to illustrate the use of the Read Map register in read mode 0.
; Animates by copying a 16-color image from VGA memory to system memory,
; one plane at a time, then copying the image back to a new location
; in VGA memory.
;
; By Michael Abrash 4/2/88
;
stack   segment word stack 'STACK'
        db      512 dup (?)
stack   ends
;
data    segment word 'DATA'
IMAGE_WIDTH     EQU     4       ;in bytes
IMAGE_HEIGHT    EQU     32      ;in pixels
LEFT_BOUND      EQU     10      ;in bytes
RIGHT_BOUND     EQU     66      ;in bytes
VGA_SEGMENT     EQU     0a000h
SCREEN_WIDTH    EQU     80      ;in bytes
SC_INDEX        EQU     3c4h    ;Sequence Controller Index register
GC_INDEX        EQU     3ceh    ;Graphics Controller Index register
MAP_MASK        EQU     2       ;Map Mask register index in SC
READ_MAP        EQU     4       ;Read Map register index in GC
;
; Base pattern for 16-color image.
;
PatternPlane0   label   byte
        db      32 dup (0ffh,0ffh,0,0)
PatternPlane1   label   byte
        db      32 dup (0ffh,0,0ffh,0)
PatternPlane2   label   byte
        db      32 dup (0f0h,0f0h,0f0h,0f0h)
PatternPlane3   label   byte
        db      32 dup (0cch,0cch,0cch,0cch)
;
; Temporary storage for 16-color image during animation.
;
ImagePlane0     db      32*4 dup (?)
ImagePlane1     db      32*4 dup (?)
ImagePlane2     db      32*4 dup (?)
ImagePlane3     db      32*4 dup (?)
;
; Current image location & direction.
;
ImageX          dw      40      ;in bytes
ImageY          dw      100     ;in pixels
ImageXDirection dw      1       ;in bytes
data    ends
;
code    segment word 'CODE'
        assume  cs:code,ds:data
Start   proc    near
        cld
        mov     ax,data
        mov     ds,ax
;
; Select graphics mode 10h.
;
        mov     ax,10h
        int     10h
;
; Draw the initial image.
;
        mov     si,offset PatternPlane0
        call    DrawImage
;
; Loop to animate by copying the image from VGA memory to system memory,
; erasing the image, and copying the image from system memory to a new
; location in VGA memory. Ends when a key is hit.
;
```

2

```
AnimateLoop:
;
; Copy the image from VGA memory to system memory.
;
        mov     di,offset ImagePlane0
        call    GetImage
;
; Clear the image from VGA memory.
;
        call    EraseImage
;
; Advance the image X coordinate, reversing direction if either edge
; of the screen has been reached.
;
        mov     ax,[ImageX]
        cmp     ax,LEFT_BOUND
        jz      ReverseDirection
        cmp     ax,RIGHT_BOUND
        jnz     SetNewX
ReverseDirection:
        neg     [ImageXDirection]
SetNewX:
        add     ax,[ImageXDirection]
        mov     [ImageX],ax
;
; Draw the image by copying it from system memory to VGA memory.
;
        mov     si,offset ImagePlane0
        call    DrawImage
;
; Slow things down a bit for visibility.
;
        mov     cx,1000h
DelayLoop:
        loop    DelayLoop
;
; See if a key has been hit, ending the program.
;
        mov     ah,1
        int     16h
        jz      AnimateLoop
;
; Clear the key, return to text mode, and return to DOS.
;
        sub     ah,ah
        int     16h
        mov     ax,3
        int     10h
        mov     ah,4ch
        int     21h
Start   endp
;
; Draws the image at offset DS:SI to the current image location in
; VGA memory.
;
DrawImage       proc    near
        mov     ax,VGA_SEGMENT
        mov     es,ax
        call    GetImageOffset  ;ES:DI is the destination address for the
                                ; image in VGA memory
        mov     dx,SC_INDEX
        mov     al,1            ;do plane 0 first
DrawImagePlaneLoop:
        push    di              ;image is drawn at the same offset in
                                ; each plane
        push    ax              ;preserve plane select
        mov     al,MAP_MASK     ;Map Mask index
        out     dx,al           ;point SC Index to the Map Mask register
        pop     ax              ;get back plane select
        inc     dx              ;point to SC index register
        out     dx,al           ;set up the Map Mask to allow writes to
                                ; the plane of interest
        dec     dx              ;point back to SC Data register
        mov     bx,IMAGE_HEIGHT ;# of scan lines in image
DrawImageLoop:
        mov     cx,IMAGE_WIDTH  ;# of bytes across image
        rep     movsb
        add     di,SCREEN_WIDTH-IMAGE_WIDTH
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

3

```
                                ;point to next scan line of image
        dec     bx              ;any more scan lines?
        jnz     DrawImageLoop
        pop     di              ;get back image start offset in VGA memory
        shl     al,1            ;Map Mask setting for next plane
        cmp     al,10h          ;have we done all four planes?
        jnz     DrawImagePlaneLoop
        ret
DrawImage       endp
;
; Copies the image from its current location in VGA memory into the
; buffer at DS:DI.
;
GetImage        proc    near
        mov     si,di   ;move destination offset into SI
        call    GetImageOffset  ;DI is offset of image in VGA memory
        xchg    si,di   ;SI is offset of image, DI is destination offset
        push    ds
        pop     es      ;ES:DI is destination
        mov     ax,VGA_SEGMENT
        mov     ds,ax   ;DS:SI is source
;
        mov     dx,GC_INDEX
        sub     al,al           ;do plane 0 first
GetImagePlaneLoop:
        push    si              ;image comes from same offset in each plane
        push    ax              ;preserve plane select
        mov     al,READ_MAP     ;Read Map index
        out     dx,al           ;point GC Index to Read Map register
        pop     ax              ;get back plane select
        inc     dx              ;point to GC Index register
        out     dx,al           ;set up the Read Map to select reads from
                                ; the plane of interest
        dec     dx              ;point back to GC data register
        mov     bx,IMAGE_HEIGHT ;# of scan lines in image
GetImageLoop:
        mov     cx,IMAGE_WIDTH  ;# of bytes across image
        rep     movsb
        add     si,SCREEN_WIDTH-IMAGE_WIDTH
                                ;point to next scan line of image
        dec     bx              ;any more scan lines?
        jnz     GetImageLoop
        pop     si              ;get back image start offset
        inc     al              ;Read Map setting for next plane
        cmp     al,4            ;have we done all four planes?
        jnz     GetImagePlaneLoop
        push    es
        pop     ds              ;restore original DS
        ret
GetImage        endp
;
; Erases the image at its current location.
;
EraseImage      proc    near
        mov     dx,SC_INDEX
        mov     al,MAP_MASK
        out     dx,al   ;point SC Index to the Map Mask register
        inc     dx      ;point to SC Data register
        mov     al,0fh
        out     dx,al   ;set up the Map Mask to allow writes to go to
                        ; all 4 planes
        mov     ax,VGA_SEGMENT
        mov     es,ax
        call    GetImageOffset  ;ES:DI points to the start address
                                ; of the image
        sub     al,al           ;erase with zeros
        mov     bx,IMAGE_HEIGHT ;# of scan lines in image
EraseImageLoop:
        mov     cx,IMAGE_WIDTH  ;# of bytes across image
        rep     stosb
        add     di,SCREEN_WIDTH-IMAGE_WIDTH
                                ;point to next scan line of image
        dec     bx              ;any more scan lines?
        jnz     EraseImageLoop
        ret
EraseImage      endp
;
; Returns the current offset of the image in the VGA segment in DI.
```

4

```
;
GetImageOffset  proc    near
        mov     ax,SCREEN_WIDTH
        mul     [ImageY]
        add     ax,[ImageX]
        mov     di,ax
        ret
GetImageOffset  endp
code    ends
        end     Start

; *** Listing 2 ***
;
; Program to illustrate use of read mode 1 (color compare mode)
; to detect collisions in display memory. Draws a yellow line on a
; blue background, then draws a perpendicular green line until the
; yellow line is reached.
;
; By Michael Abrash 4/2/88
;
stack   segment word stack 'STACK'
        db      512 dup (?)
stack   ends
;
VGA_SEGMENT     EQU     0a000h
SCREEN_WIDTH    EQU     80      ;in bytes
GC_INDEX        EQU     3ceh    ;Graphics Controller Index register
SET_RESET       EQU     0       ;Set/Reset register index in GC
ENABLE_SET_RESET EQU    1       ;Enable Set/Reset register index in GC
COLOR_COMPARE   EQU     2       ;Color Compare register index in GC
GRAPHICS_MODE   EQU     5       ;Graphics Mode register index in GC
BIT_MASK        EQU     8       ;Bit Mask register index in GC
;
code    segment word 'CODE'
        assume  cs:code
Start   proc    near
        cld
;
; Select graphics mode 10h.
;
        mov     ax,10h
        int     10h
;
; Fill the screen with blue.
;
        mov     al,1            ;blue is color 1
        call    SelectSetResetColor ;set to draw in blue
        mov     ax,VGA_SEGMENT
        mov     es,ax
        sub     di,di
        mov     cx,7000h
        rep     stosb           ;the value written actually doesn't
                                ; matter, since set/reset is providing
                                ; the data written to display memory
;
; Draw a vertical yellow line.
;
        mov     al,14           ;yellow is color 14
        call    SelectSetResetColor ;set to draw in yellow
        mov     dx,GC_INDEX
        mov     al,BIT_MASK
        out     dx,al           ;point GC Index to Bit Mask
        inc     dx              ;point to GC Data
        mov     al,10h
        out     dx,al           ;set Bit Mask to 10h
        mov     di,40           ;start in the middle of the top line
        mov     cx,350          ;do full height of screen
VLineLoop:
        mov     al,es:[di]      ;load the latches
        stosb                   ;write next pixel of yellow line (set/reset
                                ; provides the data written to display
                                ; memory, and AL is actually ignored)
        add     di,SCREEN_WIDTH-1 ;point to the next scan line
        loop    VLineLoop
;
; Select write mode 0 and read mode 1.
;
        mov     dx,GC_INDEX
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

5

```
        mov     al,GRAPHICS_MODE
        out     dx,al           ;point GC Index to Graphics Mode register
        inc     dx              ;point to GC Data
        mov     al,00001000b    ;bit 3=1 is read mode 1, bits 1 & 0=00
                                ; is write mode 0
        out     dx,al           ;set Graphics Mode to read mode 1,
                                ; write mode 0
;
; Draw a horizontal green line, one pixel at a time, from left
; to right until color compare reports a yellow pixel is encountered.
;
; Draw in green.
;
        mov     al,2            ;green is color 2
        call    SelectSetResetColor ;set to draw in green
;
; Set color compare to look for yellow.
;
        mov     dx,GC_INDEX
        mov     al,COLOR_COMPARE
        out     dx,al           ;point GC Index to Color Compare register
        inc     dx              ;point to GC Data
        mov     al,14           ;we're looking for yellow, color 14
        out     dx,al           ;set color compare to look for yellow
        dec     dx              ;point to GC Index
;
; Set up for quick access to Bit Mask register.
;
        mov     al,BIT_MASK
        out     dx,al           ;point GC Index to Bit Mask register
        inc     dx              ;point to GC Data
;
; Set initial pixel mask and display memory offset.
;
        mov     al,80h          ;initial pixel mask
        mov     di,100*SCREEN_WIDTH
                                ;start at left edge of scan line 100
HLineLoop:
        mov     ah,es:[di]      ;do a read mode 1 (color compare) read.
                                ; This also loads the latches.
        and     ah,al           ;is the pixel of current interest yellow?
        jnz     WaitKeyAndDone  ;yes-we've reached the yellow line, so we're
                                ; done
        out     dx,al           ;set the Bit Mask register so that we
                                ; modify only the pixel of interest
        mov     es:[di],al      ;draw the pixel. The value written is
                                ; irrelevant, since set/reset is providing
                                ; the data written to display memory
        ror     al,1            ;shift pixel mask to the next pixel
        adc     di,0            ;advance the display memory offset if
                                ; the pixel mask wrapped
        jmp     HLineLoop
;
; Wait for a key to be pressed to end, then return to text mode and
; return to DOS.
;
WaitKeyAndDone:
WaitKeyLoop:
        mov     ah,1
        int     16h
        jz      WaitKeyLoop
        sub     ah,ah
        int     16h     ;clear the key
        mov     ax,3
        int     10h     ;return to text mode
        mov     ah,4ch
        int     21h     ;done
Start   endp
;
; Enables set/reset for all planes, and sets the set/reset color
; to AL.
;
SelectSetResetColor     proc    near
        mov     dx,GC_INDEX
        push    ax              ;preserve color
        mov     al,SET_RESET
        out     dx,al           ;point GC Index to Set/Reset register
        inc     dx              ;point to GC Data
```

6

```
        pop     ax              ;get back color
        out     dx,al           ;set Set/Reset register to selected color
        dec     dx              ;point to GC Index
        mov     al,ENABLE_SET_RESET
        out     dx,al           ;point GC Index to Enable Set/Reset register
        inc     dx              ;point to GC Data
        mov     al,0fh
        out     dx,al           ;enable set/reset for all planes
        ret
SelectSetResetColor     endp
code    ends
        end     Start

; *** Listing 3 ***
;
; Program that draws a diagonal line to illustrate the use of a
; Color Don't Care register setting of 0FFh to support fast
; read-modify-write operations to VGA memory in write mode 3 by
; drawing a diagonal line.
;
; Note: Works on VGAs only.
;
; By Michael Abrash 4/2/88
;
stack   segment word stack 'STACK'
        db      512 dup (?)
stack   ends
;
VGA_SEGMENT     EQU     0a000h
SCREEN_WIDTH    EQU     80      ;in bytes
GC_INDEX        EQU     3ceh    ;Graphics Controller Index register
SET_RESET       EQU     0       ;Set/Reset register index in GC
ENABLE_SET_RESET EQU    1       ;Enable Set/Reset register index in GC
GRAPHICS_MODE   EQU     5       ;Graphics Mode register index in GC
COLOR_DONT_CARE EQU     7       ;Color Don't Care register index in GC
;
code    segment word 'CODE'
        assume  cs:code
Start   proc    near
;
; Select graphics mode 12h.
;
        mov     ax,12h
        int     10h
;
; Select write mode 3 and read mode 1.
;
        mov     dx,GC_INDEX
        mov     al,GRAPHICS_MODE
        out     dx,al
        inc     dx
        in      al,dx           ;VGA registers are readable, bless them!
        or      al,00001011b    ;bit 3=1 selects read mode 1, and
                                ; bits 1 & 0=11 selects write mode 3
        jmp     $+2             ;delay between IN and OUT to same port
        out     dx,al
        dec     dx
;
; Set up set/reset to always draw in white.
;
        mov     al,SET_RESET
        out     dx,al
        inc     dx
        mov     al,0fh
        out     dx,al
        dec     dx
        mov     al,ENABLE_SET_RESET
        out     dx,al
        inc     dx
        mov     al,0fh
        out     dx,al
        dec     dx
;
; Set Color Don't Care to 0, so reads of VGA memory always return 0FFh.
;
        mov     al,COLOR_DONT_CARE
        out     dx,al
        inc     dx
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Abrash Code, Continued

```
        sub     al,al
        out     dx,al
;
; Set up the initial memory pointer and pixel mask.
;
        mov     ax,VGA_SEGMENT
        mov     ds,ax
        sub     bx,bx
        mov     al,80h
;
; Draw 400 points on a diagonal line sloping down and to the right.
;
        mov     cx,400
DrawDiagonalLoop:
        and     [bx],al ;reads display memory, loading the latches,
                        ; then writes AL to the VGA. AL becomes the
                        ; bit mask, and set/reset provides the
                        ; actual data written
        add     bx,SCREEN_WIDTH
                        ; point to the next scan line
        ror     al,1    ;move the pixel mask one pixel to the right
        adc     bx,0    ;advance to the next byte if the pixel mask wrapped
        loop    DrawDiagonalLoop
;
; Wait for a key to be pressed to end, then return to text mode and
; return to DOS.
;
WaitKeyLoop:
        mov     ah,1
        int     16h
        jz      WaitKeyLoop
        sub     ah,ah
        int     16h     ;clear the key
        mov     ax,3
        int     10h     ;return to text mode
        mov     ah,4ch
        int     21h     ;done
Start   endp
code    ends
        end     Start

; *** Listing 4 ***
;
; Sample code to illustrate various approaches to generating local
; symbols within REPT loops that don't seem to work with MASM 5.0.
;
        dosseg
        .model  small
        .code

macl    macro
        rept    10
        local   testlabel
        jmp     testlabel
testlabel:
        endm
        endm

mac2    macro
        local   testlabel
        rept    10
        jmp     testlabel
testlabel:
        endm
        endm

start:
        macl

        mac2

        rept    10
        local   testlabel
        jmp     testlabel
testlabel:
        endm

        end     start
```


You've Seen all the Rest Now go for the Best!

Absolutely NO Royalties!

NOW SHIPPING Ver 2.10 with Data Entry Capabilities!

- **Create Interactive, Animated, Automated:**
  - Demos
  - Tutorials
  - User/Program Shells
  - Teaching Aids
  - Children's Education
  - Continuously Running Presentations
- **Orders of Magnitude above the rest:**
  - Can generate BASIC BLOADable Screens
  - Can generate Turbo C Source for Screens
  - Can generate ANSI files.
  - Sound and Music Capabilities
  - Mathematics and String Logic
  - User Inputs and Outputs
  - Printer Control
  - Loops, Subroutines, Decision Making
  - Screens are directly usable by Application Programs
  - Supports all 256 Colors and Characters
  - Screen Capture Program
  - Runs on CGA, EGA and Monochrome
  - Much, Much More!
- **Easy to use. Has context sensitive help.**
  - Complete, comprehensive 150+ page User Manual guides you through every step of the way.

Just $59.95 Complete!

Sample Demo Disk available for $2.00 (Credited towards your first order)

Call Today to order by credit card or send $59.95 plus $3.00 shipping/handling to:

GILMORE SYSTEMS

P. O. Box 3831

Beverly Hills, CA 90212-0831

(213) 275-8006

Requires IBM PC, XT, AT or compatible. DOS 2.0 or higher. 320 K RAM minimum. Fixed disk recommended.

Number 135 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

X Window— A View from DOS

by
M. Steven Baker

Please Pass the Windex

This is the first of a series in which I'd like to focus on technical aspects of various windowing and graphics systems from a DOS perspective. I've started with the acronym X because it has the shortest name. Actually, my hidden motive is that DOS programmers hear a lot about X Windows but rarely get a chance to see the system. Also, the X Windows system is a valuable educational vehicle for studying windowing and graphics because (1) it is fast becoming an accepted standard on workstations (which implies a modicum of success), and (2) although copyrighted, X was developed by a consortium from university and industry and is available in the public domain with a large quantity of source code.

Xtra! Xtra! Read All About It

The X Windows System (or simply X) was developed at MIT over the last several years with considerable support from industry. The primary industry supporter (with both funds and staff) has been Digital Equipment (DEC) although a number of other firms have contributed. While IBM was funding research at Carnegie Mellon (the Andrew windowing system and the MACH operating system) and supplying students with IBM RTs, DEC was busy supporting work at MIT on X and providing VAX workstations to MIT students and staff. In January 1987, eleven large computer firms including DEC, HP, Tektronix, Apollo, Data General, Sony, Adobe, and others joined with MIT in adopting X as an industry standard for graphical computing in networked environments. Conspicuously absent at the onset were Sun and IBM, who have since joined the fold. Standards are affected by politics, and the decisions to support X were likely influenced by political as much as technical motives. Sun had been very successful with its Network File System (NFS), so much so that DEC, workstation vendors, and most recently IBM have had to license this technology from Sun. If some capable alternative hadn't been developed by DEC, IBM, and the other major players, they'd have been faced with licensing Sun's windowing technology next.

X provides* hooks *(a mechanism) rather than* religion *(a policy) for interface design.

The development of X began at MIT in 1984 under the auspices of Project Athena. Project Athena and the MIT Laboratory for Computer Science all of a sudden had dozens—soon to be thousands—of workstations with bitmap displays and needed a window system to make this new computer hardware useful. Although they started with VAX hardware, they expected equal numbers of IBM workstations. Portability was thus one of the early primary goals. Although initially based on Berkeley Unix, X was conceived from the outset as a network protocol that should not depend on aspects of the operating system. The actual name X derives from a Stanford precursor called the W Windows system.

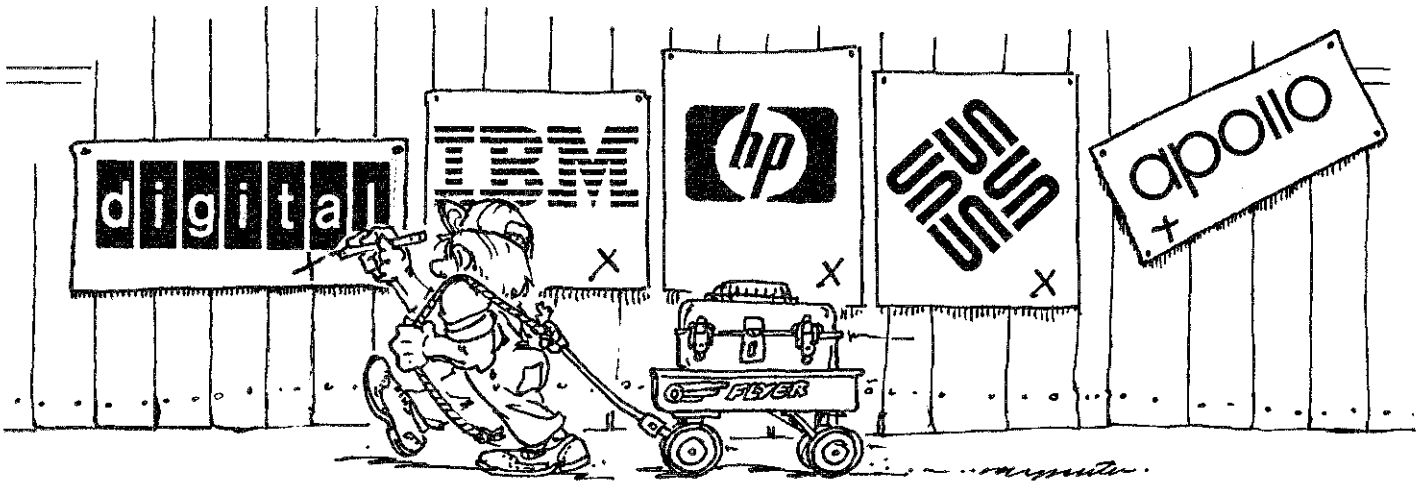
X Marks the Spot

Among the many concepts embodied in the design of X, the most important specifications set forth by the developers were the following:

- ease of implementation on a variety of displays
- device independence for application programs (that is, you shouldn't have to rewrite or recompile your application to support different display hardware)
- network transparency (an application should be able to run on one machine and be displayed on another machine)
- support for many different application and window management interfaces (X does not require a particular user interface design and should be able to support any such design)
- support for concurrent display of multiple applications
- support for overlapping windows, including sending output to partially obscured windows
- support for a hierarchy of resizable windows and for the use of several windows by an application
- high-performance, high-quality support for text, 2D synthetic graphics, and imaging (doesn't everybody want this?)
- extensibility (an extension mechanism should allow additions to the core system)

One of the more interesting concepts for PC programmers is that the windowing system should not dictate the "user interface." The X developers conceded from the start that no single user interface design is "best." Users may have very different preferences for interfaces. For example, experts and novices

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



may make very different demands on a system. And the special requirements of applications themselves may dictate different kinds of interfaces. Rather than require a specific user interface, X allows a wide range of interfaces to be constructed. X provides *hooks* (a mechanism) rather than *religion* (a policy) for interface design. The base system contains primitives from which menus and other objects may be constructed. This is certainly worlds apart from the programmer's model of Microsoft Windows.

The X System Model

X is based on a client-server model, which may be confusing jargon to a PC programmer. The X client-server relationship means that for every physical display (screen), there is a controlling server program. A client program and a server communicate via a reliable byte stream (8-bit) protocol. The client and server programs are separate and do not have to reside in the same machine. For example, the server might run on your PC and the actual application program might be running on a VAX, a Cray, or another PC. The server shouldn't care. The X protocol merely requires a reliable byte stream for communication. This could be based on local interprocess communication (server and client on the same machine) or a network connection using anything from an RS-232 line to an ethernet or token-ring network board. X is most commonly implemented on Unix machines over network connections using TCP/IP (a Department of Defense network protocol—see *PJ*, Miscellaneous, 6.2). Although a simple serial line could be used, decent response and performance mandate 50,000+ baud or more because of the quantity of client-server communication. Note that the X client-server model is just the opposite of PC file servers, where the file server is the big fast machine and the lowly PC is the client.

The X server is the workhorse of the window system. The server provides all the fundamental graphic resources and mechanisms for the system. The X server contains almost all device specific code so that all these hardware dependencies are on only one end of a network connection. The X Server directly controls the screen (display), keyboard, and mouse hardware along with other X devices such as bar code readers. Any new video or input hardware needs only the development of a new server implementation to be fully functional. The X server code itself is designed as a layer of device independent code overlying the hardware dependent device code.

I was fascinated by the possibilities for X in the evolving PC world. If I could port an X server to the PC, I could

Porting X—Tribulations of a Part-timer

I started corresponding with Project Athena staff regarding X over a year ago. At the time, the current X Windows system was Version 10 Release 4. X Version 11 was released in September of 1987, and incorporated very substantial changes from Version 10. Where Version 10R4 was a relatively austere system, Version 11 was aimed at correcting all of the deficiencies in the earlier prototypes but with increased complexity and size. While an X Version 10R4 server might take 140K, the X11 servers had grown to a characteristic 400K. Apple dropped its contract for an A/UX port of X11 after the consultant who had done a number of X10 ports had to revise his contract costs after working a few days with the new X11 distribution.

The first release of Version 11 also came with quantities of bugs (over 100 patch fix files had been issued by early 1988). I received my first distribution tape containing X11 in November 1987, free, thanks to the GNU project (Richard Stallman and Co.). Please remember that X was developed almost exclusively on machines running Unix, so the distribution came as a Unix tar tape (that is, one big file containing all the directories and files). On the GNU tape, the large X11 tar file had been compressed using the Unix compress (public domain) utility program (similar in concept to the ARC program used in the PC world).

The first step was to read the tape on the friendly minicomputer and kermit the compressed tar file (7 megabytes) to the PC—no problem, except for transfer time at 19,200 baud. The next step was to uncompress the tar file. After some searching, I located a copy of the source code to compress via my local university connections and Usenet. To the contrary of what you may read, porting programs from the 32-bit Unix world to the 16-bit DOS world is not without its frustrations. The Unix compress utility uses a 16-bit LZW encoding (ARC uses at most 12-bits) necessitating 256K (64K entries by 4 bytes) for the main array and the use of longs instead of ints. After several tries, I finally got an executable DOS version of compress, albeit agonizingly slow (several hours to uncompress a 7 megabyte file to 20 megabytes of source code)—halfway to the PC. The next step was to find a public domain version of the Unix tar program (through university connections again). After a few trials and more tribulation, I had a DOS version of tar functioning. Aside from truncated Unix file names, I finally got the X source code into the PC. Now the real work begins.—*MSB* ♦

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

develop and run client programs on the Definicon 68020 coprocessor board or the A. I. Architects 80386 Hummingboard (see *PJ* 6.1), communicating with a DOS X server via interprocess communication. On a 386 machine using one of the 386 DOS extenders, I could run the server as a DOS TSR in the lower 640K communicating with a client developed and running in the protected mode in extended memory. Using one of the newer high-end graphics boards based on the Texas Instruments 34010 processors, I could develop an X server that ran mostly on the display card and communicated with client programs on the PC. Most of these new TI 34010 based cards have 256–512K of program memory on board in which to download programs for emulation. Finally, a PC based X server could communicate with other machines using a network connection—although normally I don't have a Sun or VAX sitting conveniently down the hall.

Sample X Protocol Requests

CreateWindow	Creates an unmapped window (not viewable and assigns an identifier to it
ChangeWindowAttributes	Changes attributes of a window
GetWindowAttributes	Returns current attributes of the window
DestroyWindow	Window and all inferiors are destroyed
DestroySubwindows	Performs a DestroyWindow on all children
ChangeSaveSet	Adds or removes specified window from client's "save-set"
ReparentWindow	Changes the parent of a window and inserted as child of the specified parent
MapWindow	Window is mapped (made viewable)
MapSubwindows	Performs a MapWindow on all unmapped children
UnmapWindow	Window is unmapped (made not viewable)
UnmapSubwindows	Performs an UnmapWindow on all mapped children
ConfigureWindow	Changes the configuration of the window
CirculateWindow	Raises or lowers a mapped child in the stack of windows
QueryTree	Returns the root, parent, and children of the window
GetGeometry	Returns the root and current geometry of a drawable
InternAtom	Returns the atom (resource-ID) for a given name and creates if it doesn't exist
GetAtomName	Returns the name for a given atom
ChangeProperty	Alters the property of an atom for a specified window
DeleteProperty	Deletes the property of an atom for the specified window
GetProperty	Return the property for an atom in a window
ListProperties	Returns the atoms of properties currently defined on the window
SendEvent	Sends event to a window or possibly on to all child windows
GrabPointer	Actively grabs control of the pointer device for a window
UngrabPointer	Releases the pointer
GrabButton	Establishes a passive grab of a button/key for a window
UngrabButton	Releases the passive button/key combination
GrabKeyboard	Actively grabs control of the keyboard for a window
UngrabKeyboard	Releases the keyboard
GrabKey	Establishes a passive grab of specific key for a window
UngrabKey	Releases the key combination
QueryPointer	Returns the relative pointer coordinates to root window
GetMotionEvents	Returns all events in motion history buffer that fall between a start and stop time
TranslateCoordinates	Translates coordinates between a source and destination window
WarpPointer	Moves the pointer relative to current position
QueryKeymap	Returns a bit vector for the logical state of the keyboard indicating which keys are pressed

The Hardware It eXpects

X makes several general assumptions about the underlying hardware. It expects that some form of fast interprocess communication mechanism between the server and the client will be available. X Windows is based on modeling the display as a two-dimensional array of pixels which may have some depth (for representing color or shades of gray). Both monochrome and color displays are supported. The number of bits per pixel, and how the value translates into a color, depends on hardware. X has a most elegant model for keyboard and mouse event input. The keyboard is considered a collection of buttons that sometimes go down and sometimes go up, possibly in combination with other keys. Significance is attached to both key press and release events. A large number of modifier keys are supported along with a variety of key symbols and meanings. The mapping of any keycode is under the control of the X server. A pointing device (usually a mouse) can have buttons up to a maximum of ten although most of the applications provided with the X distribution assume a three-button mouse. X requires support for cursor, text output, and line drawing, which may all be implemented in software. X assumes very little about the underlying display hardware. Because of this design, an X server can be implemented on a simple memory mapped video board (a frame buffer in graphics jargon) such as a Hercules card or on a powerful display card with its own CPU (like the many new PC cards based on TI 34010 graphics coprocessors).

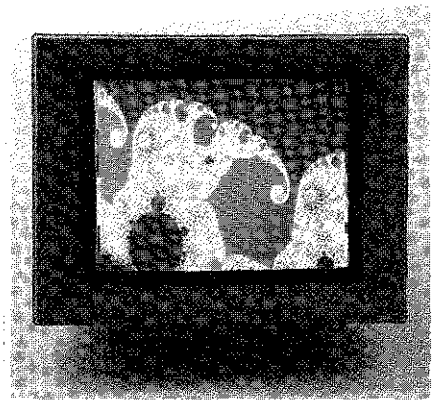
The X Protocol

I can't begin to describe the complete protocol specification here, but I'll try to give you a taste of it and refer you to the actual documentation. The first issue that the X protocol attacks is the byte ordering problem. The sending of 16- or 32-bit quantities over an 8-bit byte stream for various machines requires that some decision on byte ordering be made (the big-endian, little-endian problem). The X server is designed to accept both big-endian and little-endian connections. Client

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

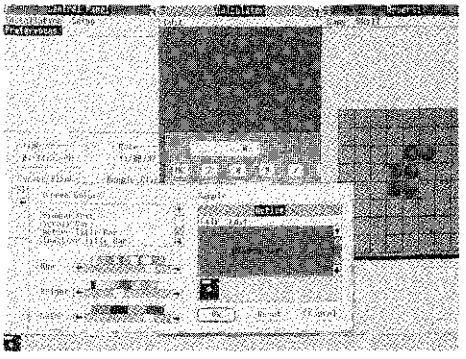
VERY HIGH RESOLUTION

The PC Tech COLOR and MONOCHROME video processor boards employ the TMS 34010 high performance graphics co-processor to insure the best possible video performance at reasonable prices.



Color 34010 Video Processor:

- **Featured on the cover of Micro Cornucopia.**
- **From 800 x 512 through 1024 x 800 resolution (depending on monitor and configuration).**
- **8 Bits per pixel for 256 simultaneous colors**
- **Hardware support for CGA/MDA emulation.**
- **PC, XT, and AT compatible**

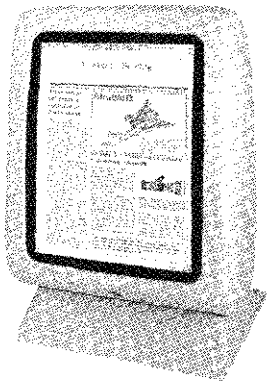


The PC Tech Color 34010 video processor is a superior 34010 native code and DGIS development tool. We support up to 4 megabytes of program (non-display) 34010 RAM as well as up to 768K bytes of display RAM. ***Compare our architecture and prices to any other intelligent graphics board. Then choose the PC Tech Color 34010 Video Processor for your development engine and your production requirements as well.***

Color 34010 Video Processor $1,195.00

Price includes 512K display RAM, 1024K program RAM, and utility software. Monitor not included.
Also available: DGIS, 34010 C compiler, assembler, 34010 fractal software,
additional display and program memory, and various monitor options.

PC Tech Monochrome 34010 Video Processor and Monitor



- **736 x 1024 resolution (other options available)**
- **2 bits per pixel for 4 hardware gray shades**
- **Hardware support for CGA/MDA/Hercules emulation**
- **PC, XT, and AT compatible**
- **Full page 66 line text editing with many popular editors**
- **Excellent windows 2.0 application development system**

The graphics and bit manipulation capabilities of the TMS 34010 make the PC Tech Monochrome 34010 Video Processor 66 line full page text and graphics display faster than many 25 line systems. The video processor is available separately or with the high resolution white phosphor monitor shown above.

Monochrome 34010 Video Sub-System $1,295.00

Price includes Monochrome Video Processor and monitor pictured above.
Also available: DGIS, TI 34010 C compiler, TI assembler.
Monochrome 34010 Video Processor also available separately.

Designed, Sold and Serviced By:

PTech

904 N. 6th St.
Lake City, MN 55041
(612) 345-4555
(612) 345-5514 (FAX)

Number 208 on the Reader Service Card

PC, XT, AT, DGIS, Hercules, and Windows 2.0 are trademarks or registered trademarks of their respective companies.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

programs always send and receive in their native byte order (except that raw bitmaps and image data are sent to the server in its preferred format). The server is responsible for all byte swapping. The X protocol also takes care of word alignment problems on various architectures. The protocol always consists of blocks that are multiples of 32 bits, and each 16-bit and 32-bit quantity within a block is aligned on 16-bit and 32-bit boundaries, respectively. A protocol request from a client is simply a data packet containing an opcode, some fixed length parameters, and a variable amount of data.

The X protocol supports four formats: (1) request, (2) reply, (3) error, and (4) event. Every request format contains an 8-bit "major" opcode and a 16-bit length field expressed in units of 32-bits (4 bytes) followed by additional bytes of data. The length field defines the total request length, including the header. The actual maximum block size to use for communication is determined by the server at connection setup and can range from a minimum of 16K to a 256K maximum. The base X protocol uses opcodes through 127, and opcodes 128 through 255 are reserved for extensions. Every request on a given connection is implicitly assigned a sequence number starting with 1 that is used for replies, errors, and events. The client sends request formats to the server. The side bar that accompanies this article provides a sample list of some requests that are supported.

Reply formats (from the server) are similar, each containing a length field expressed in 4 byte units along with any data bytes. Every reply also contains the least significant 16 bits of the sequence number of the corresponding request. Reply formats are a minimum of 32 bytes long.

Error formats (from the server) are 32 bytes long. Every error packet includes an 8-bit error code, where again error codes 128 to 255 are reserved for extensions. Every error packet also includes the opcode of the failed request and the least significant 16 bits of the sequence number of the request.

Event formats (keyboard or mouse input sent from the server to the client) are 32 bytes long. Every event contains an 8-bit type code where event codes 64 through 127 are reserved for extensions. An event packet also contains the least significant 16 bits of the sequence number of the last request issued by the client. An event packet also contains a millisecond timestamp field set by the server.

The X graphics model is completely pixel or raster based. The core protocol supports displays with pixels up to 32 bits in depth. This should satisfy your urge for a few more colors on your CGA board. Request opcodes support graphics primitives such as point, line, polyline, and polygon fills along with much higher level abstract calls. Request opcodes also support resource IDs for types WINDOW, PIXMAP (an image bitmap of off-screen memory), CURSOR (which may be of any size or shape), FONT (supports both fixed width and proportional fonts stored as bitmaps), GCONTEXT (graphics context of defaults for various values), and COLORMAP. Clients request creation of a resource by supplying various parameters (for example the name of a font). The server allocates the resource and returns a resource-ID (often called a handle in the DOS world) passed as a unique 32-bit value. Any client that knows the identifier of a resource can use and manipulate the resource at will—even if the resource was created by another client. This capability allows window managers to be written independent of other applications. For example, to display some text in a window, the parameters passed in a request packet would include the drawing color and the identifiers for the window and the font, along with the actual string of characters.

X supports an arbitrary branching hierarchy of rectangular windows. At the top is the root window, which covers the entire screen. Windows are visually clipped by their parent windows. The coordinate system has an origin (0,0) in the upper left window corner with *x* axis horizontal and *y* axis vertical. This corresponds directly with most hardware. A window can optionally have a border. Windows can be created, destroyed, mapped and unmapped, moved and resized, and raised to the top or lowered to the bottom of a stack. X supports

Figure 1: Connection Setup

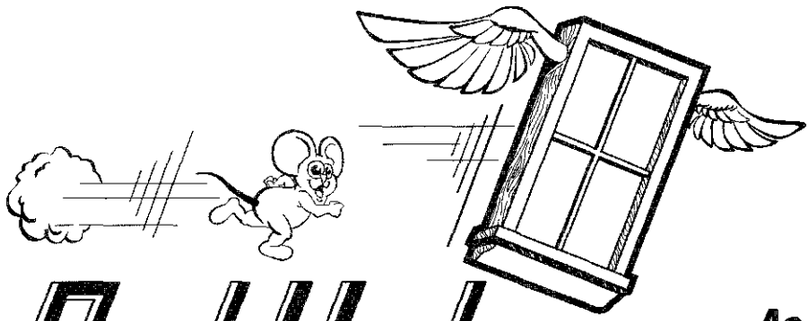
Client sends the following:	
byte-order byte	(either "B" or "l")
protocol-major-version	unsigned 16-bit
protocol-minor-version	unsigned 16-bit
authorization-protocol-name	string
authorization-protocol-date	string
Received by the Client	
success:	byte (true or false)
protocol-major-version:	unsigned 16-bit
protocol-minor-version:	unsigned 16-bit
length:	unsigned 16-bit
Additional data received if authorization fails:	
reason:	string
Additional data received if successful:	
vendor:	string
release-number:	unsigned 32-bit
resource-id-base and mask:	unsigned 32-bit
image-byte-order:	LSBfirst or MSBfirst
bitmap-scanline-unit:	8, 16, or 32
bitmap-scanline-pad:	8, 16, or 32
bitmap-bit-order:	Least or Most Significant
pixmap-formats:	list of supported FORMATs
roots:	list of possible SCREENs
motion-buffer-size:	unsigned 32-bit (for mouse events)
maximum-request-length:	unsigned 16-bit (packet max)
min-keycode, max-keycode:	for the keyboard

Note: SCREENs and FORMATs are further data structures. For example, a SCREEN data structure contains information about the various display screens supported by the X server. This includes screen width and height in pixels and millimeters, the allowed pixel depths, default colormap data, the value to be used for a white-pixel and a black- pixel, installed pixel maps, whether the server supports off screen memory to save over-written screen real estate, etc.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ATTENTION BASIC PROGRAMMERS

"... you should give serious thought to using QuickWindows."
Charles Olsen, Computer Shopper, June '88



QuickWindows ADVANCED

As seen in the Microsoft Value Pack catalog.

Finally, an easy to use Window Management Library loaded with features that work in both text and graphics modes. Hundreds of hours went into the development, testing and debugging of QuickWindows Advanced. The original QuickWindows has proven to be a popular library among several Fortune 500 companies, International users, and governmental organizations both here and abroad.

SPECIAL OFFER

for Programmers Journal readers:

SAVE $25.00

off the regular price of QuickWindows Advanced.

Offer ends 8-31-88

☑ **Yes, please send me QuickWindows Advanced for only $114 (US), regularly $139. Add $5 shipping.**

☐ CHECK ☐ VISA ☐ MC

_____ Exp _____

Name _____

Address _____

City _____

State _____ Zip _____

Phone _____

OFFER ENDS AUGUST 31, 1988. VALUE PACK REBATE VOID WITH THIS OFFER

You can easily create:

- Windows, Pop-up and Pull-down menus
- Scrollable and Indexable Help Windows
- Dialog Boxes with Buttons, Checkboxes, Rangebars, Horizontal and Vertical Listboxes, and Input Fields
- Pop-up Message Boxes with Pushbuttons
- Data Entry Screens
- Character Fonts (8x8 through 8x24)
- A Window for DOS/BIOS

Supports:

- All text and graphics modes through BASIC SCREEN 13 on MDA, CGA, EGA, EEGA, VGA, and MCGA cards
- Text modes 132 x 44 on ATI VIP and other similar cards
- Microsoft Compatible Mouse
- QuickBASIC 3.0/4.0 and BASCOM 5.36/6.0

Use in Programs Developed For:

- Business
- Scientific and Engineering Community
- Medical Profession
- Music Industry
- Personal Use

QuickWindows Advanced includes:

- Library disk with 90 functions CALLable from BASIC
- 130+ page (well-documented) manual
- Several example programs
- Demo/tutorial program
- 30-day money-back guarantee
- Extended Technical support
- Free access to our 24-hr BBS

To place your order or to request a Free demo disk call:

(401) 274-5465

Visa, MC, COD accepted.

Software Interphase INCORPORATED

5 Bradley Street, Suite 7 • Providence, RI 02908

Number 115 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

several window types: (1) opaque (the most common), (2) transparent (always invisible but provides clipping), and (3) input-only.

Resources have associated with them interesting properties such as bit-gravity (if a window is resized, in which direction this drawable should be moved), window-gravity, copy-from-parent (whether class, pixel depth, etc., should be copied from the parent), along with defaults for borders, background pixmaps, cursors, fonts, colormaps, etc. Bit image data may be sent as either single bit-planes or pixel-depths.

For remote clients, the X protocol can be built on top of any reliable byte stream. A connection setup process begins with a client program's sending an initial byte to identify the byte order to be used. An ascii "B" means that values will be transmitted most significant byte first while an ascii "l" means least significant byte first. With the exception of image data, all 16-bit and 32-bit quantities sent by the client and those returned by the server must use this byte order. Following the byte-order byte, the client sends a block of information identifying the protocol version and revision numbers and authorization information if security is implemented. The server sends back a Boolean success or failure block with similar information. If the connection was successful, the server also appends to this block a large list of detailed information about the server. This process is conceptually illustrated in Figure 1.

The X Distribution

The first release of X11 in September 1987, contained over 20 megabytes of source code. X Version 11 Release 2 was made available on March 1988. This second release of Version 11 comes with over 60 megabytes of source, mostly in C. Release 2 cleaned up a lot of bugs from the earlier release and is much more robust. No changes have been made to the X protocol and other basic components. The current release (X11 Release 2) is divided into two distributions—supported (the basic X stuff) and contributed (user contributed code). Each distribution contains two Unix **tar** files—one for source code and one for printable versions of the documentation, mostly in Postscript (PS) format ready to be dumped to a laser printer. If downloaded directly over Arpanet from MIT, the four **tar** files are compressed and split into parts of 512,000 bytes each. The following table summarizes the **tar** file sizes:

Distribution	Contents	Megabytes	Needed
supported	source code	20	required
supported	documentation (PS)	7	optional
contributed	source code	22	recommended
contributed	documentation (PS)	10	optional

The X distribution has grown in size not because of changes to the underlying protocol, but because more sample servers are included, more X toolkits are incorporated, and more user contributed programs are bundled. The distribution depends on extensive makefiles and a few utilities to be constructed for your environment. X server code is included for the machines and displays from DEC, Sun, Apollo, HP, the IBM RT, and sample monochrome and color servers. The new IBM servers include source code support for the EGA and VGA, along with the older display hardware available on the RT (IBM also included a binary object module supporting the 8514 display). The default distribution will build easily only on a Unix system (preferably Berkeley 4.2 or better) for the supported hardware. A C compiler that has 32-bit integers will also make the porting job easier. While X Version 10R4 was a more spartan system, Version 11 was intended to correct all of the deficiencies in the earlier prototypes but added complexity. While an X version 10R4 server might take 140K, the X11 servers have grown to a typical 400K.

XLIB, Toolkits, and Clients

The X distribution also includes an XLIB library in C, encapsulating the basic X protocol operations as several hundred C function calls. A draft Common Lisp interface library is also included. These libraries should be considered similar to the low level I/O calls available for Unix. The distribution also includes four higher level toolkits designed to make developing applications easier (these may be thought of as the STDIO package under Unix). The four toolkits provided are described below:

X Toolkit Intrinsics: provides a collection of resource, event, and object managers written in C that may be used to build user interface objects called "widgets." A sample set of widgets is provided.

Xrlib: provides a collection of user interface utilities and higher-level objects written by Hewlett-Packard. Future releases are to be based on the X Toolkit Intrinsics.

Andrew Toolkit: developed by IBM and Carnegie Mellon for the Andrew System at CMU, this package provides a collection of integrated, high-level objects and applications.

Interviews: developed at Stanford, this package provides tools for building user interface objects in the C++ programming language.

The distribution also contains a large number of client programs. These range from the simpler bouncing ball demos, clocks, and calculator programs to terminal emulators, mail programs, editors, and a window manager. The earlier release contained several window managers. Again, one of the strengths of X is that it doesn't dictate a window manager: One could write a window manager that looked like the MAC, Microsoft Windows, or the new Open Look windowing manager specification under joint development by AT&T and Sun for Unix.

Last eXit

I've tried to provide a brief introduction to X and convey some of its flavor. The size and breadth of X is representative of the new, more powerful windowing and graphics systems—that is, large amounts of code initially containing hundreds of bugs. In the case of X, we are fortunate to have the source code readily available for review and study. The X development is now managed by the X Windows consortium, a group of industry organizations and MIT.

In closing I've included the ubiquitous Hello World program, this version written by David Rosenthal of Sun Microsystems using the X toolkit. ♦

PRODUCED 2023 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Permission to use, copy, modify and distribute this document for any purpose and without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies, and that the name of Sun Microsystems, Inc. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

Specification

The "Hello World" problem should be slightly restated for the world of window systems. The program needs to

- Create a window of an appropriate size and position it on the screen if required.
- Paint the string "Hello World" in a suitable font, centered in the window.
- Paint the string in a color that will contrast with the window background—it is not acceptable to paint an invisible string.
- Deal with its window being exposed—repainting the window if required.
- Deal with its window being resized—re-centering the text.
- Deal with its window being closed into an icon and re-opened, providing suitable identification of the icon.

Note that this specification is much more complex than the canonical "Hello World" program, so it's likely that the resulting programs will be significantly more complex.♦

Copyright 1987 by Sun Microsystems, Inc.

```
#include <stdio.h>
#include <X11/Xlib.h>
#include <X11/Intrinsic.h>
#include <X11/Atoms.h>
#include <X11/Label.h>

#define STRING  "Hello,  World"

Arg wargs[] = {
XtNlabel,    (XtArgVal) STRING,
};

main(argc, argv)
int argc;
char **argv;
{
    Widget      toplevel, label;

/*
* Create the Widget that represents the window.
* See Section 14 of the Toolkit manual.
*/
    toplevel = XtInitialize(argv[0], "XLabel", NULL, 0, &argc, argv);

/*
* Create a Widget to display the string,  using wargs to set
* the string as its value.  See Section 9.1.
*/
    label = XtCreateWidget(argv[0], labelWidgetClass,toplevel, wargs,
XtNumber(wargs));

/*
* Tell the toplevel widget to display the label.  See Section 13.5.2.
*/
    XtManageChild(label);

/*
* Create the windows,  and set their attributes according
* to the Widget data.  See Section 9.2.
*/
    XtRealizeWidget(toplevel);

/*
* Now process the events.  See Section 16.6.2.
*/
    XtMainLoop();
}
```

The X Windows 11 Release 2 distribution is available for downloading via Arpanet or may be purchased from MIT on two 9-track Unix *tar* tapes at $200 each (one tape for the Core distribution and one for the User Contributed software). Documentation is included when you purchase the magnetic tapes. If you just want more details, printed documentation is also available from MIT in two sets: (1) X Core Distribution (800+ pages) at $30; and (2) X User-Contributed Software (500+ pages) at $20. Tapes or documentation may be ordered prepaid from MIT.

MIT Technology Licensing Office
77 Massachusetts Ave., Rm E32-300
Cambridge, MA 02139
(617) 253-6966

I'd like to thank several hardware and software vendors for product loans I'm using to port X Windows to the PC:

Texas Instruments (Austin, TX): TMS34010 Development Board, 34010 C compiler, and assembly language tools.
PC Tech (Lake City, MN): TMS34010 Color Graphics Board
Logitech (Fremont, CA): EGA & Mouse (EGA and mouse card with multiscan monitor)♦

LALR generates parsers

for ADA, BASIC, C, Pascal, Modula 2, SQL, dBASE, C++, 386 Assembly, FORTRAN...

LALR 3.0 is a complete LALR(1) parser generator. It's fast, powerful, and easy to use. So, if you're developing a compiler, translator or language interface, you want **LALR**.

If you just want to learn about language translation and state-of-the-art parsing technology, you want **LALR**.

"unbelievably fast ... can handle very large grammars"
COMPUTER LANGUAGE MAG., DEC. 1985

LALR 3.0 includes:
- Grammars for ADA, BASIC, Turbo Pascal and Turbo C.
- Parser skeleton source code with error recovery written in C language.*
- Lexical scanner, syntax checker and calculator source code in C.

60-DAY MONEY-BACK GUARANTEE

$99

714-832-LALR

LALR Research • 1892 Burnt Mill • Tustin, CA 92680

* Parser skeletons may be written in other languages.
Requires: DOS 2.0 or later, 256K or 512K for large grammars. Overseas orders: Add $10.00.

Number 188 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

A Digital Clock for Microsoft Windows

by
Michael J. Babcock
William S. Hall

In this article, we show you how to write a simple program in Microsoft Windows. When all's done, you will have a useful utility containing the essential ingredients of every Windows program that can serve as a model for your independent efforts. At the same time you will see a variety of techniques commonly used in Windows—registering a window class, borrowing data from a previous instance, reading the windows initialization file, creating a window and sizing it in a device-independent way, creating a logical font, using the timer, painting the window, and segmenting a program. If you understand these ideas, you will be able to write your own elementary applications and you will have a basis for further study.

This program is written for the Windows 2.0 and Windows/386 environments. Windows 1.0 is already obsolete, and if you are a registered owner of Windows, you should take advantage of the $25 2.0 upgrade offer from Microsoft.

Code for the program is available on the *PJ* bulletin boards and Issue Disk for Volume 6.4. Also included on the BBS's and disk are the small memory model template described at the end of the article and a useful device driver, OX.SYS, which we have found to be immensely valuable in analyzing and debugging Windows programs.

Description of the Program

DC (Digital Clock) shows the system time in a window. If DC is running as an overlapped window, the time in hours, minutes, and seconds is redisplayed about once every second. If the window is iconic, the time in hours and minutes is shown in the icon area, but updates occur only once a minute. What is interesting about the icon display is the need to load and use a small font so that all the information can be seen. As it is difficult to predict the readability of the icon display for a given video adapter, we have given the user the option of selecting the font's parameters using the Windows initialization file **win.ini**. The parameters default to those we have found to be satisfactory for a display comparable to the EGA's.

The **win.ini** entries (which you should make using a text editor such as NOTEPAD, the Windows version of EDLIN) are as follows:

```
[DC]
fontheight=n1
fontwidth=n2
fontweight=n3
```

where n1, n2, and n3 are the height, width, and weight of the font. Experimentation suggests that 8, 6, and 700 are good values for an EGA, VGA, or Olivetti/AT&T 640 x 400 screen whereas 6, 4, and 400 produce good results on the CGA.

The first time DC is run, its window is placed in the lower right corner of the screen. Each subsequent instance appears at a location selected by Windows. If you enter DC.EXE in the **load=** line in **win.ini**, the program will start up as an icon when you execute Windows.

Windows Programming Software

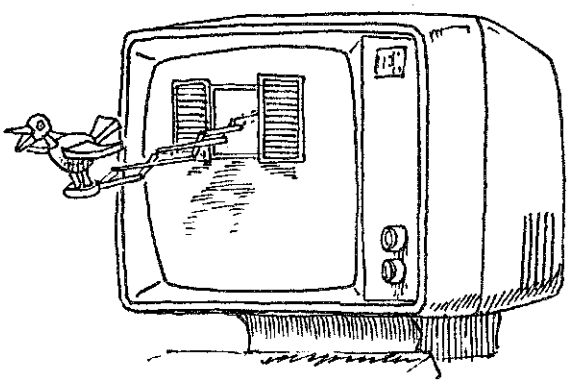
You'll need the Microsoft C compiler, Version 4.0 or greater. As far as we know, no other current C compiler is "Windows aware" although Microsoft has freely distributed the specifications. Certain Windows functions require a special prologue and epilogue, and the Microsoft compiler is able to furnish this code. You must also have the Windows Software Development Kit (SDK), where you will find the libraries, headers, and utilities you need in order to write programs in Windows. Although each package lists for about $500, they are frequently heavily discounted to around $270 each. If you already own earlier versions of either the compiler or SDK, by all means upgrade them.

Windows Programming Hardware

If you have a machine capable of running Windows, you have the basic hardware needed to develop Windows programs. But you may not be very well prepared to debug them. Because Windows owns all the machine's resources, you need to have an auxiliary terminal connected to one of the communications ports. Then you can run SYMDEB (supplied with the SDK) and direct all debugging input and output to the terminal. You can even debug at the source code level if you've set the right switches during compilation and linking; however, you can't see automatic variables unless you understand how the stack is arranged. Despite these and other limitations, SYMDEB can be a useful tool for tracking problems in Windows.

You can use temporary **printf** statements for debugging purposes (even though the output will not show on the display) if you run Windows with output redirected to an appropriate device such as an auxiliary terminal, printer, or file. Standard output will then appear there, and this can be very helpful in analyzing what a Windows call is ac-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



tually doing. But don't leave any permanent **printf**s in your program.

As an alternative to the auxiliary terminal, you may want to run dual monitors using an EGA, CGA, or VGA as the primary video and a monochrome or Hercules board as the secondary display. If you combine this arrangement with the public domain program OX.SYS, you'll have a very useful arrangement. By redirecting Windows to AUX, you can see standard output displayed on the monochrome screen. SYMDEB can be redirected there as well, and you do not lose the use of a communications port.

Windows Program Memory Models

Because Windows is a multitasking environment, several programs may be in CPU memory at the same time. Lacking a real paging system, Windows takes advantage of the segmentation architecture of the Intel processor. Most Windows programs are made up of several segments, each located in its own area of the global heap. These segments can be moved, shared, and even discarded. And although the effects of memory management can generally be ignored by the programmer, there are times when he or she must be aware of the consequences. For example, a long pointer to a segment that can be moved will not be valid indefinitely. And if the program's code is monolithic, Windows can't make efficient use of the limited memory available. Thus, it really is necessary to plan a program's optimal layout.

One of the simplest schemes for taking advantage of Windows memory management is to divide a program into two segments, one of which contains only initialization code. If the initialization segment is so marked, it will be discarded when it is no longer needed and its space will be made available for other programs. Since at least two code segments are required, the so-called medium memory model is implied. In fact, this is the best model for writing Windows programs having only one data segment simply because it makes the mechanics of program segmentation very easy.

It is also possible to have more than one code segment in the small memory model as we have done here, But certain precautions must be observed. For example, calls between different segments must be labeled FAR. You can call a library function only from the _TEXT segment (the default code segment) since all such functions are NEAR. **WinMain**, the entry point for Windows, must be in _TEXT. Finally, unless you say something specific about the segments in the program's definition file, they will be grouped together by the linker anyway. We will explain this in greater detail below.

A window class is basically a data structure containing a collection of attributes common to all windows of that class.

The Make File for DC

Windows programs have so many pieces that you really must use some kind of **make** program. Such a tool is supplied with the Microsoft C compiler. Its syntax is similar to the Unix version's and so will be familiar to many of you. We have tried to keep ours simple, but we have taken advantage of the macro feature in order to help you accommodate a variety of compiler versions and libraries. You can use these macros to tailor the **make** file to your own environment.

Beginning at the top of Listing 1, you will see some compiler macros with the switches needed for making "normal" as well as debugging versions. Although you should review the exact meanings of the switches used with the compiler, here is a summary of the ones we use:

-d	show each pass of the compiler
-c	make an object file only
-AS	compile for the small memory model
-Gw	compile for Windows (PASCAL keyword)
-Zp	pack structures
-Ox	maximum optimization

You can't omit **-Gw** or **-Zp**. In the debug compile macro, the optimization has been removed and an additional switch **-Zd** has been added to place line number information in the object file. By default, stack probes are restored.

Then follow several macros that show you which libraries to use at link time. The macro you should choose depends on your compiler version. We have also assumed that you have *not* elected to make compound libraries if you have either the OS/2 development kit or the 5.0 compiler. If you have built composites, please develop your own library macro; there are too many variations for us to guess your configuration.

If you plan to use floating point numbers, read how to do this in the Windows SDK. You must avoid linking with the emulator library.

After the macro definitions is the command needed to make the resources for the program. Resources are objects such as dialog boxes, fonts, strings, icons, bitmaps, and menus. A source resource .RC file describes these objects in text format using the appropriate syntax. The resource compiler RC then converts this description into a binary

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

.RES file. After the executable .EXE file is made, RC is again used to attach the .RES file to the .EXE. DC has only one resource, which is not typical of a Windows program, but we wanted to keep it simple.

Next are the commands to make the .OBJ files from the .C sources. (To make a debugging version, replace the **cp** macro with **cpd**.) You should note the named segments specified by the -NT switch. This information is used by the linker in combination with the .DEF file to separate the code into two segments.

Finally, the Windows executable file is put together using several commands. Note the use of a special linker, LINK4. Windows programs use the same .EXE format as OS/2. Some of the new features provided are dynamic linking (linking to libraries at runtime rather than at link time) and memory management (some aspects of which we have already described).

The **align** command is not critical; it just specifies the separation of segments in the disk file. The default is 512, so the file is a bit smaller if 16 is used. We have also included the line number switch **li** in case you have compiled for debugging. The appropriate library macro, which you should select according to your configuration, follows. Then comes a new element, the definitions file, which we examine next.

After the executable is made, the resources must be attached. Again, the resource compiler is run, this time without the **-r** switch. Then a symbol table file that can be read by SYMDEB is created by the MAPSYM utility.

The Definitions File

Every Windows program requires a .DEF file. Ours (Listing 2) is minimal. The first entry is the name of the program. You *must* be sure that it agrees with the program's .EXE name; otherwise, multiple instances will not be recognized and code segments will not be shared.

The description string is optional, but the stub is rather interesting. This is the MS-DOS program you want to be executed if someone tries to run your program outside Windows. Generally, WINSTUB.EXE, which is supplied on the Windows SDK, is used, but it could just as well be Lotus 123, or perhaps more realistically, a non-Windows version of your program.

Next is a general declaration about the code and data. Since you can have several instances of the same program in memory, data is not shared and thus must have as many realizations as you have instances. Code, however, can be shared, and this is the default. We allow both code and data to be moveable.

We also supply specific information about the two code segments we have created. This information is *absolutely necessary* if you want to have two separate segments during runtime in the *small* memory model. Note that we have marked the initialization segment as discardable.

We have exported the only entry point Windows will have into our program, the window procedure **MainWndProc**. Windows will now be able to locate and call this function whenever it has information for us. Note that *all* functions in a program that Windows may call—dialog box handlers, callback functions, and window procedures—must be exported in the application's .DEF file.

The Resource File

DC's only entry in the resource file (Listing 3) is a string table. Although a Windows program can have strings as part of the source code, it is usual to put them into the resource file. Since the resources are also in segments that are moveable, discardable, and separate from the rest of the code or data, it is advantageous in terms of runtime memory considerations to keep strings here.

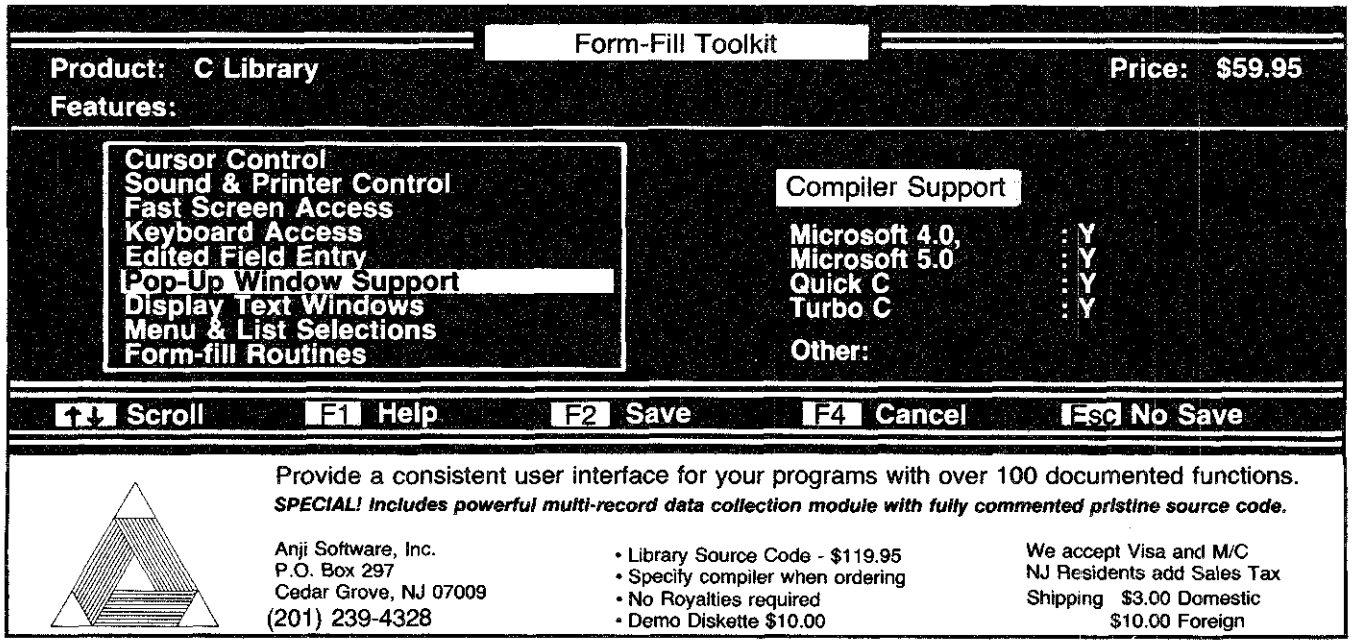
In addition, it's easy to internationalize a program by translating the strings and then replacing the resources with the new ones. It's not necessary to recompile the entire program but only to pass it through the resource compiler.

In our resource file we keep the strings used for the application name, part of the time display, reading **win.ini**, and getting information about the display adapter on which Windows runs.

Associated with each string is a constant defined in the program's include file. When the string is needed, it is loaded by referring to this value.

The Header File

The header file (Listing 4) is quite simple. Here we declare those variables shared by the two code segments, and we define the string manifests. By use of the device at the top of the file, we ensure that the globals are declared only in the file where EXTERN is defined as



Number 162 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

null while in all other files, globals are noted as external.

We also declare the two functions referenced in both modules since their forms differ from the small memory model default's. In particular, the initialization function, which is an inter-segment call, is declared FAR, and the windows procedure, since it is called by Windows, is tagged as FAR and PASCAL. This last keyword means that variables are pushed on the stack from left to right, an order opposite from that of standard C. The prologue/epilogue code is added, and finally, the callee, not the caller, restores the stack. All Windows functions follow this convention, but functions local to a program do not have to be declared PASCAL.

The Initialization Module

Now we can discuss the code. Look at the DCINIT.C module (Listing 5). At the top, note the inclusion of **windows.h**, a file supplied on the Windows SDK. Everything needed by a Windows program—data structures, type definitions, macros, constants, and all Windows function declarations—is defined in **windows.h**. You can find many undocumented features of Windows here. Although **windows.h** is very large, it is possible to substantially reduce compile time by inhibiting certain definitions. The technique is described at the beginning of the file.

Below the include statements and the local function declarations, you will see **InitProgram**, the first function called when DC is invoked. The parameters passed are the same as those given to **WinMain**, the program's entry point. Two of these are instance handles; the first to the current instance, the second to a previous invocation of the program. There is also a long pointer to the command line, but this string is not broken into its fields as is **argv** in C. The parameter **cmdShow** indicates how the window is to be initially displayed—iconic, normal, or maximized (zoomed).

If this is the first invocation of our program, we load those strings used in the time display, register the class of the main window, and create a logical font small enough to be used when writing to the icon. If this is a later instance, we simply borrow these strings from some previous instance. We can also retrieve a handle to the small font. It is not necessary to register the window class again nor to recreate the font.

Next, we create and display a window of the class we have just registered. Then we start the Windows timer, causing a message to be sent to our Windows procedure about once per second. We use this as a signal to read, record, and if necessary, update the time display.

The timer can be used in one of two ways. In this program we simply ask for a WM_TIMER message to be placed in our message queue whenever a timer event occurs. By using different parameters in **SetTimer**, we can request Windows to directly call a procedure of our own, a so-called "callback" function. The callback function must have a specific form, be declared PASCAL, and be exported in the definitions file. In addition, a pointer to the function must be prepared using the Windows function **MakeProcInstance**. Many operations in Windows depend on this callback principle.

If any of these actions fails, FALSE is returned to the caller and the program exits. Now, let's study more closely each routine called by the initialization function.

A window class is basically a data structure containing a collection of attributes common to all windows of that class. Included are the mouse cursor style, menu, class name, background brush, class style, and (most important) the window procedure. To register a class, it is necessary only to fill in the WNDCLASS structure and pass it to Windows. Since Windows copies it, we can destroy the structure after **RegisterWindowClass** returns successfully.

We begin by allocating space in the local heap to hold the WNDCLASS structure. Allocations in the heap can be moved, compacted, or locked in position, and Windows manages this area in somewhat the same way it does global memory. So the first step is to obtain a handle to a memory block with **LocalAlloc**. Then when the handle is to be used, it must be locked in order to retrieve its pointer. Later, when the memory is to be deallocated, it must be unlocked, then freed.

When we use the attribute LPSTR in the **LocalAlloc** call, we ensure that the memory retrieved is set to zero and is fixed. Although the fact isn't documented, the handle returned in this case is the actual pointer, and a call to **Local-**

FULL AT&T C++: ANNOUNCING VERSION 1.2

Guidelines announces its port of **version 1.2** of AT&T's C++ translator. As an object-oriented language, C++ includes: classes, inheritance, member functions, constructors and destructors, data hiding, and data abstraction. "Object-oriented" means that C++ code is more readable, more reliable and more reusable. And that means faster development, easier maintenance, and the ability to handle more complex projects. C++ is **Bell Labs' answer to Ada and Modula 2.** C++ will more than pay for itself in saved development time on your next project.

C++

from GUIDELINES for the IBM PC: $295

Requires IBM PC/XT/AT or compatible with 640K and a hard disk.
Note: C++ is a *translator*, and requires the use of Microsoft C 3.0 or later.

Here is what you get for $295:
- The full AT&T v1.2 C++ translator.
- Libraries for stream I/O and complex math.
- **The C++ Programming Language,** the definitive 327-page tutorial and description by Bjarne Stroustrup, designer of C++.
- Sample programs written in C++ .
- Improved installation guide and documentation.
- 30-day money-back guarantee.

To Order:
send check or money order to:

GUIDELINES SOFTWARE, INC.
P.O. Box 749, #PJ
Orinda, CA 94563

To order with Visa or MC, phone (415) 254-9183.
(CA residents add 6% tax.)

C++ is ported to the PC by Guidelines under license from AT&T.
Call or write for a free C++ information package.

Number **204** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Lock isn't really necessary. But we go through the formalities to illustrate proper techniques for working with the heap.

Now we fill in the WNDCLASS structure for the clock program. We use a standard arrow cursor. Since we have no bitmapped icon nor menu, we enter a NULL for each. As the class name should be unique, we use the name of our program. Because we want Windows to maintain the background, we specify a brush (which in this case is white). (If an application wants to paint its own background, a NULL is entered in the WNDCLASS background brush field, but the application must then be prepared to create its own brush and to respond to the WM_ERASEBKGND message from Windows.) Next we supply the instance handle, and we specify in the style field that we want the window to be redrawn if it is resized either vertically or horizontally. Finally, we name a long pointer to **MainWndProc**, which Windows will call when it has a message for the application.

There are two other fields in the structure that we have left at zero, namely the class- and window-extra data. By asking for space here, the programmer can store useful information common to the class or unique to each window in the class.

If the registration is successful, we then unlock and free the memory used and return TRUE.

In the registration procedure, we obtained the class name by reading it from the resource segment. If this is a second or later instance of the program, it is not necessary to register the class again. Nevertheless, we still need the application name as well as several other strings in the program, and we want a handle to the special font we require. We load this data from a previous instance of the program.

Now we are ready to create the window. If this is the first instance, we want the window to show up in the lower right hand corner of the screen. In addition, we require that the window be large enough to display one line of text of the form

```
hh:mm:ss A.{P.}M.
```

This string requires 13 characters. Clearly, we must be able to create a window with a client area (the region in which the application can draw) one character high and at least 13 characters wide. In addition, we have to measure the screen size (in pixels) so that we can properly place the window of the first instance. To obtain this information we need a display context for the screen. Normally, a window display context is obtained from the window handle, but as we have not created the window, the handle is not yet available. But we can always get an "Information Context" for the display. We can then read the size of the default font with the **GetTextMetrics** call and obtain the essential screen data with **GetDeviceCaps**. After that, the information context can be deleted.

The next step is to find out how big the Window must be in order for the client area to have the desired size. To this end we fill out a rectangle with the desired location and size of the client and call **AdjustWindowRect** to get the actual window rectangle needed; however, because we have set the client area to be at the upper left corner of the screen, the coordinates of the window's upper left corner will be negative. This may explain the peculiar mathematics involved in actually calculating the values of the variables **width** and

If you see something here you like, by all means, compare it with what our competition has to offer.

Seidl Version Manager (SVM) Version 2.0 is a comprehensive source code configuration system designed for professional software developers. SVM tracks changes made to individual modules (revisions) and stores "version" information that tells which revisions go into making which versions of a product. SVM supports multiple developers in both networking and non-networking environments. Some of its features include: revision branching, revision merging, user ids and optional passwords, seven different file access levels, a menu driven shell, formatted reports with user definable layouts, central journal logging, on-line help, optional data compression, a remote site update utility, and more. ***$299.95, LAN version $1000.00***

Seidl Make Utility (SMK) Version 2.1 is the most sophisticated product generation utility available. SMK is not just another copy of the of the UNIX make. SMK uses a structured, high-level, *dependency definition language (DDL)* that is a programming language in itself. It supports symbolic constants, parameterized macros, for loops, if-then-else style conditionals, interactive statements, ambiguous path name expressions, and much more. In addition, SMK is lightening fast and won't bog down, even on your largest projects. ***$99.95, LAN version $500.00 (includes SMKgen).***

New!* SMKgen Version 1.0** is an automatic dependency generation utility. By examining the files that make up a product, SMKgen determines how those files depend on each other and amazingly writes a dependency file that can be processed by SMK. SMKgen can also construct link files and *filelist* files that integrate with SVM. With SMKgen, the task of placing a software project under SMK is reduced to executing a single command. ***$49.95

"SVM is a full-featured system that has all the essential capabilities you are likely to need, and is further distinguished by several important features that none of its competitors yet have." ***- The C Users Journal™, February, 1988.***

When you know the facts, you'll know we're the best.

For more information call: **1-313-662-8086**

For a free copy of our competitive comparison report, write to us on your company letterhead. MC/VISA/COD accepted.

SEIDL COMPUTER ENGINEERING

3106 Hilltop Dr., Ann Arbor, MI 48103

Number **174** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

height, which are then passed to the **CreateWindow** call.

To generate a window of the class we have just registered, we have to supply Windows with additional information through the **CreateWindow** call. The window title is the same as the application's, which is just the class name. The style, which we have already defined when adjusting the client rectangle, is typical of a program's top level window. The only deviation from the standard WS_OVERLAPPEDWINDOW style is in not allowing a MAXIMIZE box.

Although we want to place the first instance of the window in a particular place, all subsequent instances need to appear somewhere else. (Otherwise, you will see only the most recent instance as one will be superimposed upon the other!) So we simply accept the default location by entering CW_USEDEFAULT,0 for the coordinates of the upper left corner. But we do use the width and height so laboriously computed above. Since a top level window has no parent, we need no menu, and since we have no special data required during creation, these parameters are set to NULL.

If this is the first instance of the program and the creation is successful, we measure the window's actual size, which may differ from the requested value since most windows have a set of minimum dimensions. We combine the values found from **GetWindowRect** with the screen dimensions already found and move the window to the lower right corner of the screen.

Note that this code works for any display. Windows offers many calls to allow the programmer to write code in a device-independent way.

Finally, the window is displayed and updated. **UpdateWindow** is a request for Windows to send the application a WM_PAINT message. In many Windows programs, all output to the screen is done in response to this message. Usually, a program generates this output by (1) modifying some data structure used by its paint routine to draw on the screen and (2) marking as "invalid" the corresponding rectangle on the screen where this data will appear. Windows then sends a paint message, and the application updates the screen. Of course, an application can write to the screen whenever a display context can be obtained, but the indirect method just described is quite common, and we make extensive use of it in this program.

Currently, the icon window on a high resolution display such as the EGA is usually 36 pixels square and the standard font is 8 pixels wide and 10 pixels high. On a VGA or AT&T/Olivetti monochrome screen, the font is taller but has the same width. So if the system font is used, it would be impossible to show three lines, one of which has five characters, without clipping. But Windows has a variety of fonts that can be used. By describing the desired font in a logical font structure, we may be able to obtain one that is suitable. This is the purpose of **GetFontInfo**.

Because it is difficult to anticipate every device, we have made the width, height, and weight of the font selectable via the Windows initialization file **win.ini**. If the user has made the entries outlined in the "Description of the Program" earlier in this article, the **GetProfileInt** call will return the values specified. If they are not found, the default values will be used.

A complete discussion of the various parameters of the LOGFONT structure and how Windows makes the selection would take us far afield. You may want

The DBMS for *TOUGH* Programmers

You are an experienced application developer. A specialist. You have worked hard to get here. And you are looking for a DBMS that will maximize your expertise. You know that all the popular DBMSs are designed for the masses — anyone can use them. You know that there are no miracles: a DBMS easy enough for anyone to use ***cannot*** also be the right one for you.

Meet ***The Andsor Collection***: the only DBMS that does not waste your talents. Many find it difficult. Because it was designed for experts. It uses an unusual concept. A combination of built-in operations and procedural language. An environment where the application logic and data are integrated into one unified structure. It baffles novices, but is incredibly powerful in the hands of experts. If you can tame this power, a rich reward awaits you: ***fully customized applications with one tenth the effort***. You deserve it. Because you have worked hard for this expertise.

- □ An interactive environment where you can build and modify an application ***even while it is running***.
- □ Programs ***one tenth*** their size in other systems.
- □ Windows, screens, help, menus, with practically no code at all.
- □ A new way of relating files, can simplify applications by creating relations that are impossible in other systems.
- □ Variable code that changes at run time, lets you perform more than one operation in the same program section.

If you are an average programmer, use the popular DBMSs: they have sold millions, and you need this comfort. ***But if you think you are a tough programmer, accept this challenge***. Write or call for our brochure and ***FREE*** demo disk. It contains two executable, real-life applications, also showing design and programming details. There are comments and help, but no reference manual. Study the applications. Then, if you can understand more than half, congratulate yourself: you are ready to step up to the one DBMS that rewards expertise.

The Andsor Collection™

60 DAY MONEY BACK GUARANTEE $145 *after July 31, 1988* $295 Visa, MasterCard, AmEx, Check

ANDSOR®
ANDSOR RESEARCH INC.

390 Bay Street, Suite 2000
Toronto, Ontario M5H 2Y2
(416) 245-8073

To order call toll free (U.S. and Canada) **1-800-628-2828 Ext. 535**

Price includes shipping in the U.S. and Canada. Please add $10 for shipping to other countries. If you return the software, $8 will be deducted from the refund, to cover our shipping cost.

System requirements: any IBM PC or PS/2 or fully compatible, 320K RAM, one drive or hard disk, monochrome or color

© 1988 Andsor Research Inc. Andsor is a registered trademark and The Andsor Collection is a trademark of Andsor Research Inc.
IBM is a registered trademark and IBM PC, PS/2 are trademarks of IBM Corporation

Number 191 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

vLIB

the user interface library for C

vLIB is a comprehensive, easy to use library of over 175 custom C functions for building sophisticated PC applications.

Windows • *overlapping, tiled, built-in window management*

Menus • *vertical, horizontal pulldown, popup, arrays*

Forms • *full screen or popup data entry with multiple field types*

Pop Ups • *messages, prompts, selection lists, scroll bars*

Formatted Screen Output

Full Editing Input

High Speed Display

... and much, much more!

Give your programs a professional look! Develop applications faster!

Compilers supported: Microsoft C and Quick C, Borland Turbo C, Lattice C, and Datalight C.

All memory models supported.

Demo disk with usable sample programs and source (includes shipping) . $5

Library with reference card and 280 page programmer's manual $99

Library (as above) plus full source code $149

$5 shipping and handling. California residents please add 7% sales tax.

No royalties. Site license available.

Call or write:

Pathfinder Associates
291 Madrone Avenue
Santa Clara, CA 95051
(408) 984-2256

Visa and MasterCard accepted.

BBS: (408) 246-0164 (1200/2400 N-8-1)

Number 145 on the Reader Service Card

to experiment with some of the fields and observe the results. If you are curious, you can read the face name of the font you have found by getting a display context, selecting the font into it, and calling **GetTextFace**.

The Resident Module

At last, we can consider the main code segment (Listing 6). We begin by defining EXTERN so that all global variables are declared in this module. Following are the declarations of two static structures. The first is needed to read the time. The second is a rectangle the coordinates of whose lower right corner are updated each time the window is moved. This rectangle always contains the dimensions of the client and is used to center text in the window.

The entry point for all Windows programs is **WinMain**. We have already described the parameters passed to **WinMain**. As you can see, ours simply calls the initialization function and then falls into the so-called message reading loop used in one form or another by every Windows program. The program stays there until **GetMessage** returns zero. **WinMain** then returns, causing the program to exit.

A typical non-Windows program gets input by querying the operating system. A windowing system, by contrast, waits for information to arrive in the form of messages. These are retrieved until the application's message queue is empty. Control then passes to another program, and the application remains suspended until a message is again available. If a message is retrieved, it is translated (an action that converts WM_KEYDOWN and WM_KEYUP messages to WM_CHAR messages and which is analogous to obtaining an ASCII value from a scan code) and then dispatched to the application's window procedure. Windows then calls the window procedure that carries out the appropriate task and returns.

Note that if an application doesn't read messages, no other process can run. Although multitasking, Windows is not preemptive. In addition, the **GetMessage** call does not return until a message is available, and it may appear that no Windows program can carry on lengthy operations in its Windows procedure. In fact, there are methods to maintain tasking using **PeekMessage**. In the future, we will try to give some examples of this kind of message handling.

Over a hundred different kinds of messages can be sent to a program. Mouse movement, mouse button clicks, keyboard action, windows movement and sizing, requests for painting, etc.—all generate messages. Each contains data in the word parameter (**wParam**) and long parameter (**lParam**), whose meanings are unique to the message. Most messages are never specifically acted upon by a program and are simply returned for default processing. In our program, we take special action on only five messages.

A WM_CREATE message is received when the corresponding window has been created but not yet displayed. This is an appropriate opportunity to perform certain initializations. Here we use it to read the time because as soon as the window shows itself it will receive a WM_PAINT message, as well. As this is the signal to display the time, it is necessary to have current data already available.

The WM_DESTROY message is the signal that the application has received a request to exit. The timer is killed, and a request is made to Windows to post a quit message to the program. When the quit message is received, **GetMessage** will return zero in the message loop and the program will exit.

A WM_SIZE message is sent whenever the window changes size. **lParam** contains the new client width and height in its respective low and high word. As we are using the window client rectangle to center text when we display the time, we need to know when this information changes.

We have set the timer to send us a WM_TIMER message about once a second. When the message arrives, we get the time. If there is an hour change, we invalidate the client area of the window and request that the background be repainted as well. If we are iconic, we record the time but do not invalidate the window client area unless the minute has changed. If not iconic, the rectangle is invalidated since we are showing the seconds, as well. Note that in the latter case we request that the background *not* be updated in order to avoid the unpleasant flashing this can cause on the screen.

The **InvalidateRect** call causes a WM_PAINT message to be placed in the message queue. The first action an

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LIBRARY SUPPORT.

ANOTHER CAREER OPPORTUNITY FOR PCs *WITHOUT* OUR 386 MOTHERBOARD.™

***With* Our MotherBoard, Your PC, XT or AT Can Run New OS/2 Releases!** If you'd rather keep your computer on your desk than under foot, get our 386 MotherBoard. Our Board will give your spreadsheet, complex CAD and graphics, and multitasking applications a boost to boast about. In fact, our line of MotherBoards accelerate your PC, PC/XT, PC/AT or clone to speeds equal to the IBM PS2 Model 80. And faster. Because we've built in 1 Megabyte of high speed RAM and a 387 math coprocessor socket.

Accelerator Cards Have No Future. An accelerator card can only turn an obsolete computer into a faster obsolete computer. Our MotherBoards advance your computer into a new 386 system! They run OS/2, Windows/386, UNIX V and PC-MOS/386. For more power, you'll find 16-bit expansion slots that accommodate the latest I/O expansion cards. **Only our 386 MotherBoards give your *PCs an unlimited future!***

The Critics Applaude! *PC Magazine* awarded our Board "The Editor's Choice" for 386 Replacement Boards. *PC World* called it "the Upgrade Product of the Year."

Technical Features ▪ 16 MHz 80386 ▪ 1 Megabyte of 100 nsec 4-way interleaved RAM ▪ PC/AT compatible I/O and BIOS for support of OS/2 ▪ Six 8-bit expansion slots ▪ two 16-bit expansion slots; four on 386 MotherBoard/AT ▪ One 32-bit expansion slot for up to 12 Megabytes of high speed memory ▪ Battery-powered clock calendar.

386 MotherBoard/PC or MotherBoard/XT	$1495
386 MotherBoard/AT	$1595
32-bit RAM Board (2 Mbyte installed; up to 4 Mbytes)	$495
16 MHz 80387 math coprocessor	$695
16-bit combination hard disk/floppy disk controller	$245

Step up your PCs with our easy-to-install 386 MotherBoards. For more information, call: **1 (800) 443-6284**. In New York: **(516) 434-1600**. All our products are available at your local computer dealer!

Hauppauge Computer Works, Inc.
175 Commerce Drive,
Hauppauge, New York 11788

Hauppauge!

Number **122** on the Reader Service Card

Trademarks: IBM PC, XT, AT and PS2: IBM.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

application takes is to retrieve a paint structure using the **BeginPaint** function. This is passed to our paint routine, where all updating is performed. The **EndPaint** call terminates a paint message. Even if an application has no local paint function, these two function calls must be executed whenever a paint message is received.

Included in the paint structure passed to **MainWndPaint** is a handle to a "Display Context," which must be obtained from Windows before any output can be performed. A display context is basically a data structure describing the objects to be used when drawing on the output device. The device itself can be a printer, the screen, or even a memory bitmap. A display context has a number of default characteristics such as a black pen, white brush, system font, text mapping mode, etc. If the application wants something different, such as a color brush or a different font, these must be selected into the display context. So one of our first actions is to see if we are iconic. If so, the small font, whose handle we retrieved during initialization, is selected; otherwise, we accept the system font.

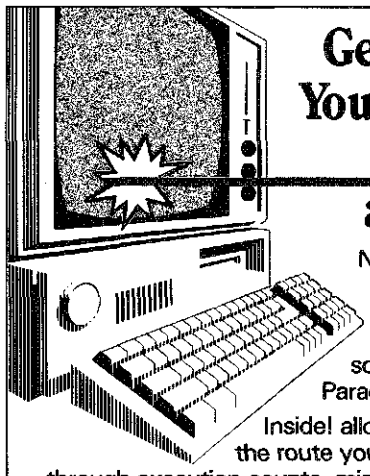
Most of the remainder of this last routine is straightforward. The last step is finally to draw the text using the client rectangle of the window. Although many different styles are possible, the ones we have specified suffice for either the icon or normal window.

DC As a Template

Now you see how a simple Windows program is constructed. It is not very easy to write even a small one, but a lot of the work is simply boilerplate. For example, every program has to register its window class and create one or more windows, read messages, and respond to a selected subset of them. Everything else is an enhancement and extension of this basic pattern tailored to the overall purpose of the program.

For example, DC can be stripped down to form a template that you can use as a basis for your own programs. Here is a suggested list of steps:

1. Remove all strings from the resource file except the application name.
2. Change the description string in the .DEF file to something more appropriate.
3. Delete all string constant definitions in the header file except the first. Also, remove all external variables except the handle to the main window and the application name string.
4. In **dc.c**, delete the static variables at the top and in the window procedure. Remove the case statements for WM_TIMER, WM_CREATE, and WM_SIZE. Delete the timer call in WM_DESTROY, all references to **MainWndPaint**, and the procedure itself.
5. In **dcinit.c**, take out **GetFontInfo** and any references to it. Remove the **LoadString** and **SetTimer** calls in **InitProgram**. Load only the **AppName** string in **GetPreviousInstanceData**. Finally, in **MakeAndShowMainWnd**, delete all code prior to the **CreateWindow** call. Substitute WS_OVERLAPPEDWINDOW for style and CS_USEDEFAULT,0 for width and height. And eliminate the special positioning code used for the first instance.
6. Make and test the result. If the program runs and supports more than one instance, you can then rename all



Get To Know Your Programs Inside! and Out!

Now you can analyze your programs with unprecedented detail with Inside!, a new software package from Paradigm Systems.

Inside! allows you to examine the route your programs take through execution counts, minimum, maximum and total elapsed times and a count of how many times each source line executes—function by function—for popular Borland and Microsoft languages like Turbo Pascal and Quick C!

New Product Offer:

Call Paradigm Systems before Sept. 1 and get your easy-to-use Inside! software for only $65.00. Inside!, which is also available in other languages, will sell for $75.00 after this special introductory offer.

PARADIGM SYSTEMS INCORPORATED

Paradigm Systems Incorporated
P.O. Box 152 Milford, MA 01757
To Place Orders (800)537-5043 Product Support (617)478-0499

Turbo Pascal is a registered trademark of Borland international Inc. Microsoft is a registered trademark of Microsoft corporation. Inside! is a trademark of Paradigm Systems Incorporated.

Number 195 on the Reader Service Card

SCREEN MANAGER
For C, BASIC, FORTRAN, COBOL Programmers

The Screen Generator ™

The Screen Generator offers professional screen management for creation of Window, Help, Menu, and Data Entry Screens for over 20 compilers. You create screens using SG's Screen Painter. All screen text and field information is stored in compact screen libraries. All Screen Display and Data Entry functions are controlled from within your program. SG's function calls provide all the flexibility and speed you'll ever need for screen display and block mode or field level data entry. The Screen Generator includes: The Screen Painter/Print Utility, The Screen Manager, and 100 page User Guide.

So give us a call! You'll be up and running in no time, while our 60-day Money Back Guarantee eliminates all the risk!

The Screen Generator ™ — New Version	
C, BASIC, ASM & Turbo Pascal	$79.
FORTRAN or COBOL	add $20.
Demo Disk Available	VISA/MC

The West Chester Group
P.O. Box 1304, West Chester, PA 19380
(215) 644-4206

Number 190 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

That's right. 4 times faster.

Clocking in at over 75,000 lines per minute on a 6MHz IBM AT, OPTASM is four times faster than Microsoft's MASM 5.0. **4 times faster —** that's 400% more throughput!

But speed is only one part of it. OPTASM is nearly 100% compatible with MASM 5.0 (except 386 support).

It is the only single assembler capable of supporting the various incompatibilities between MASM 3, 4 & 5. That makes OPTASM **more MASM compatible than any single version of MASM!**

Other features? OPTASM generates smaller code without ever generating extra NOP's. It automatically handles jumps out of range, up to 15,000 symbols and most of MASM's phase errors. It also boasts a built in MAKE and simplifies segmentation.

That's why we can make our OPTASM challenge: Test OPTASM head to head against MICROSOFT MASM 5.0. Order both assemblers with their 30-day guarantees. In a lot less than 30 days, you'll see just how dazzling OPTASM's speed really is. You'll realize that we're compatible, easier to use, and deliver many more important features than MASM. So accept our challenge. Try both assemblers. **Four times faster and more features, too.** We know which one you'll send back.

Write or call us to order or for our detailed brochure.

SLR SYSTEMS

1622 N. Main Street
Butler, PA 16001
412-282-0864
BBS 412-282-2799
Telex 559215
800-833-3061

OPTASM: $195 Guaranteed returnable within 30 days.

Number 165 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

VEDIT PLUS

#1 PROGRAMMABLE EDITOR

Call 1-800-45-VEDIT for FREE Fully Functional Demo Disk

Stunning speed. Unmatched performance. Total flexibility. Simple and intuitive operation. The newest VEDIT PLUS easily satisfies the most demanding computer professional.

Try a Dazzling Demo Yourself.

The free demo disk is fully functional—you can try all features yourself. Best, the demo includes a dazzling menu-driven tutorial—you experiment in one window while another gives instructions.

The powerful "macro" programming language helps you eliminate repetitive editing tasks. The impressive demo/tutorial is written entirely as a "macro"—it shows that no other editor's "macro" language even comes close. And VEDIT PLUS is only 40K in size.

Go ahead. Call for your free demo today, You'll see why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Only VEDIT PLUS is this Flexible.

The installation lets you pick from closely emulating the keyboard layout of Word Perfect, WordStar and others. Or you can easily create your own layout and even your own editing functions. Supports any screen size—you pick screen colors and attributes.

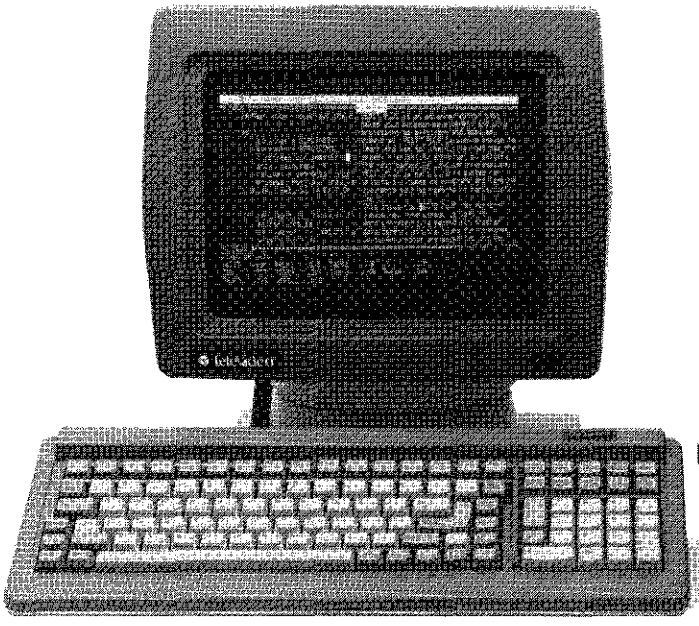
Supports the IBM PC, XT, AT and PS/2. Also supports MultiLink, PC-MOS/386, Concurrent DOS and most networks. Also available for MS-DOS, FlexOS (protected mode), CP/M-86 and CP/M. (Yes, we support windows on most CRT terminals, including CRTs connected to an IBM PC.) Order direct or from your dealer. $185.

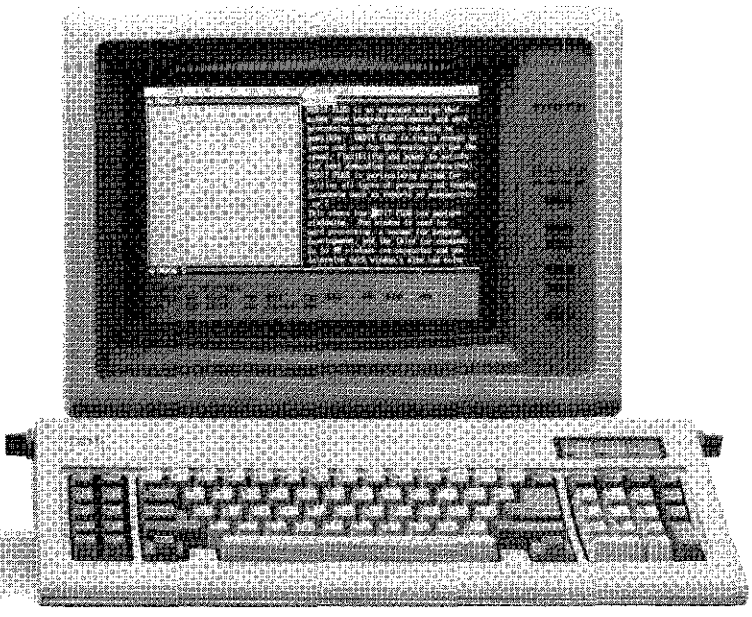
Special: VEDIT (single file, no windows) for CP/M—$49.

- **Fully Network Compatible**
- **Call for XENIX-286 version**
- **30 Day Money-back guarantee**

Compare Features and Speed

	BRIEF	Norton Editor	PMATE	VEDIT PLUS
'Off the cuff' macros	No	No	Yes	Yes
Built-in macros	Yes	No	Yes	Yes
Keystroke macros	Only 1	No	No	Unlimited
Multiple file editing	20+	2	No	20+
Windows	20+	2	No	20+
Macro execution window	No	No	No	Yes
Pop-up menus	No	No	No	Yes
Execute DOS commands	Yes	Yes	Yes	Yes
Automatic processing of Compiler errors	Yes	No	No	Yes
"Cut and paste" buffers	1	1	1	36
Undo line changes	Yes	No	No	Yes
Paragraph justification	No	No	No	Yes
Convert to/from WordStar	No	No	No	Yes
On-line calculator	No	No	No	Yes
Configurable Keyboard	Hard	No	Hard	Easy
43 line EGA support	Yes	No	No	Yes
Manual size/index	250/No	42/no	469/Yes	380/Yes
Benchmarks in 120K File:				
2000 replacements	1:15 min	34 sec	1:07 min	6 sec
Pattern matching search	20 sec	Cannot	Cannot	2 sec
Pattern matching replace	2:40 min	Cannot	Cannot	11 sec





VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PMATE is a trademark of Phoenix Technologies Ltd. Norton Editor is a trademark of Peter Norton Computing Inc. MultiLink and PC-MOS/386 are trademarks of the The Software Link, Inc. CP/M and FlexOS are trademarks of Digital Research. MS-DOS is a trademark of Microsoft.

*Also available for TI Professional, Tandy 2000, DEC Rainbow, Wyse WY700 and others.
*Demo disk is fully functional, but does not readily write large files.

CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103
(313) 996-1299, TELEX 701821

Number 56 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

files to something more appropriate, say SMLTPL*.*, making sure that the corresponding name changes are also made in the source, definition, make, and resource files. You now have a segmented, small-model template that you can use for your future efforts. ♦

Michael J. Babcock has directed electronic and computer music facilities while teaching electronic music composition as well as acoustics, audio, and traditional music theory. Mr. Babcock programs music synthesis/MIDI applications with Atari/GEM and serves on the ANSI X3H3 Computer Graphics Standards Committee. He is currently a Windows Systems Engineer for Olivetti in Cupertino, California.

William S. Hall holds degrees from the University of Virginia, Cambridge University, Brown University, and the University of Michigan. He taught college mathematics and worked for three years as an associate editor of Mathematical Review. *He is a Senior Engineer at Olivetti Advanced Technology Center in Cupertino, California, where he specializes in Windows system support software.*

Code follows.

LAHEY SETS NEW FORTRAN STANDARDS

LAHEY PERSONAL FORTRAN 77 **$95**
Low cost, Full 77 Standard, Debugger, Fast Compilation

F77L FORTRAN LANGUAGE SYSTEM **$477**
For porting or developing, this is the critics' choice.
"Editor's Choice" . . . . . ***PC Magazine***
". . .the most robust compiler tested." . . . . . ***Micro/Systems***
". . .the most efficient and productive FORTRAN development tool for the DOS environment" . . . . . ***BYTE***

F77L-EM/16-bit $695 **F77L-EM/32-bit $895**
Break through the DOS 640K barrier. The most powerful PC Fortran Language Systems for downloading or writing large programs.

PRODUCTIVITY TOOLS
Profiler, ULI Mathematical Functions Library, Overlay Linker, Toolkit, Utility Libraries, Windows, Memory Boards, 80386 HummingBoard.

IF YOU DEMAND THE VERY BEST, THEN YOU SHOULD BE USING LAHEY. CALL US TO DISCUSS YOUR PC FORTRAN NEEDS.

CALL FOR NEW FEATURES INCLUDING MATH COPROCESSOR EMULATION

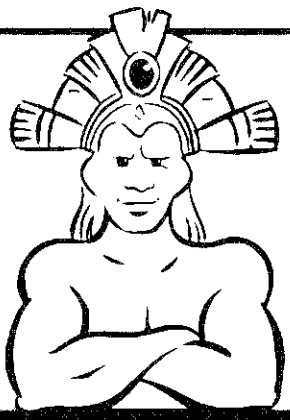
30 DAY MONEY-BACK GUARANTEE

FOR INFORMATION OR TO ORDER:
1-800-548-4778
Lahey Computer Systems, Inc.
P.O. Box 6091, Incline Village, NV 89450
TEL: 702-831-2500 TLX: 9102401256
FAX: 702-831-8123

Lahey Computer Systems Inc.

Number **44** on the Reader Service Card

EVERLOCK



PROTECTION

Designed for user-transparency, clone compatibility & strength. It features:
- no need for damaged media or I/O plugs
- supports all Hard & Floppy disk formats
- file-server networks supported
- variable number of installs (0-99)
- create demos with remote unlock option
- allows protected upgrades by modem BBS
- limit program use by date or count

$195 Starter Kit or $495 with NO meter counts. Free info & demo disk available

Az-Tech Software, Inc.
305 East Franklin
Richmond, MO 64085
(800) 227-0644 (816) 776-2700

Number **51** on the Reader Service Card

EASY FAST PROFESSIONAL

DATA ENTRY
WINDOWS
MENUS
HELP

GW Basic Basic 86 BASICA
Quick Basic Turbo C
MS Cobol MS C Turbo Basic
Lattice C
If you are serious about programming
PLEASE try HI-SCREEN XL!
Quick C
Instant C

HI-SCREEN XL™

$149 only
Multilanguage support
No Royalties
30-day risk free

C Basic dBXL Clipper
RM Cobol Fox Base
Realia Cobol Turbo Pascal
Mark Williams C
Call now for demo and information:
1-800-338-2852
in CA: (415) 397-4666
RM-Fortran
Microfocus Cobol
MS MASM dBase II, III, and III+ Turbo Prolog
MS Pascal MS Fortran MS Basic Quicksilver

"You may like other screen management tools, but you will love HI-SCREEN XL."

Softway, Inc., 500 Sutter St., Suite 222, San Francisco, CA 94102

Number **110** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available on 360K, MS-DOS disk, $10.00 postpaid. ORDER BY PHONE — VISA/MC orders call (503) 747-0800. Mail order information on page 76.

Code from "Digital Clock" by Babcock & Hall

1

Listing 1

```
## DC - Windows Digital Clock
# Created by Michael Babcock and Bill Hall
#

## some macros
# no debug compile
cp=cl -d -c -AS -Gw -Ox -Zp

# debug compile
cpd=cl -d -c -AS -Gw -Zpd

# Microsoft C 5.x compiler with OS2 libraries
LIBOS2=slibw slibc slibc3 libh/NOD/NOE

# Microsoft C 5.x compiler with PM libraries
LIBPM=slibw slibcr libh/NOD/NOE

# Microsoft C 5.0 compiler for MS-DOS
LIB5=slibw slibc libh/NOD/NOE

# Microsoft C 4.0 compiler for MS-DOS
LIB4=slibw slibc libh/NOD

## make the .RES file
dc.res : dc.rc dc.h
    rc -r dc.rc

## make the .OBJ files
dc.obj : dc.c dc.h
    $(cp) -NT _TEXT dc.c

dcinit.obj : dcinit.c dc.h
    $(cp) -NT _INIT dcinit.c

## create the .EXE file, add resources, and make a map file
dc.exe : dc dc.def dc.res dc.obj dcinit.obj
    link4 dc dcinit,dc/align:16,dc/m/li,$(LIBPM),dc
    rc dc.res
    mapsym dc
```

Listing 2

```
NAME    DC

DESCRIPTION 'Windows Digital Clock by Mike Babcock and Bill Hall'

STUB 'Winstub.exe'

HEAPSIZE        1024
STACKSIZE       4096

CODE    MOVEABLE
DATA    MOVEABLE MULTIPLE

SEGMENTS
    _TEXT PRELOAD MOVEABLE
    _INIT PRELOAD MOVEABLE DISCARDABLE

EXPORTS
    MainWndProc @1
```

Listing 3

```
#include "dc.h"
```

2

```
STRINGTABLE
BEGIN
    IDS_APPNAME,        "DC"
    IDS_DISPLAYNAME     "DISPLAY"
    IDS_TIMESTR         "TIME"
    IDS_AM              "A.M."
    IDS_PM              "P.M."
    IDS_FONTHEIGHT      "fontheight"
    IDS_FONTWIDTH       "fontwidth"
    IDS_FONTWEIGHT      "fontweight"
END
```

Listing 4

```
/*
 * Header file for the digital clock.
 *
 * Written by Michael Babcock and Bill Hall
 * April 20, 1988
 *
 */

/* This trick insures that globals are declared in only one place */
#if !defined(EXTERN)
#define EXTERN extern
#endif

/* Declarations which need to be known in all C source files */
LONG FAR PASCAL MainWndProc(HWND,unsigned,WORD,LONG);
BOOL FAR InitProgram(HANDLE, HANDLE, LPSTR, int);

/* Resource string constants */
#define IDS_APPNAME     100
#define IDS_DISPLAYNAME 101
#define IDS_TIMESTR     102
#define IDS_AM          103
#define IDS_PM          104
#define IDS_FONTWIDTH   105
#define IDS_FONTHEIGHT  106
#define IDS_FONTWEIGHT  107

/* Global variables */
EXTERN HWND hWndMain;           /* main window handle */
EXTERN HFONT hFont;             /* handle to font used when drawing in icon */
EXTERN WORD ClockTimerEventID;  /* timer event identifier */
EXTERN char szAppName[10];      /* name of the application */
EXTERN char szTimestr[10];      /* icon title string */
EXTERN char szAM[10];           /* A. M. string */
EXTERN char szPM[10];           /* P. M. string */
```

Listing 5

```
/*
  Initialization segment for the digital clock.
  This code can be discarded after it is used.
 */

#include <windows.h>
#include "dc.h"

/* local function declarations */
static BOOL NEAR RegisterWindowClass(HANDLE);
static HFONT NEAR GetFontInfo(HANDLE);
static void NEAR GetPrevInstanceData(HANDLE);
static BOOL NEAR MakeAndShowMainWnd(HANDLE, HANDLE, int);

/* This routine is FAR since it is called from another segment */
BOOL FAR InitProgram(hInstance,hPrevInstance, lpszCmdLine, cmdShow)
HANDLE hInstance, hPrevInstance;
LPSTR lpszCmdLine;
int cmdShow;
{

  /* if this is the first instance of the program ... */
```


PRODUCED 2005 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
    if (!hPrevInstance) {
      /* read some strings from the resources */
        LoadString(hInstance,IDS_TIMESTR,(LPSTR)szTimestr,sizeof(szTimestr));
        LoadString(hInstance,IDS_AM,(LPSTR)szAM,sizeof(szAM));
        LoadString(hInstance,IDS_PM,(LPSTR)szPM,sizeof(szPM));
      /* register the window */
        if (!RegisterWindowClass(hInstance))
            return FALSE;
      /* get a font for displaying the time in the icon window */
        if ((hFont = GetFontInfo(hInstance)) == NULL)
            return FALSE;
    }
  /* A previous instance already exists so get global data from there */
    else
        GetPrevInstanceData(hPrevInstance);

  /* Create and show the window */
    if (!MakeAndShowMainWnd(hInstance,hPrevInstance, cmdShow))
        return FALSE;

  /* Set the timer to generate a WM_TIMER message about once a second */
    if (!(ClockTimerEventID = SetTimer(hWndMain, NULL, 975, NULL)))
                /* at 1000 msec, clock skips a second every 24 seconds;   */
                /* at  975 msec, clock repeats a second every 87 seconds  */
        return FALSE;

    return TRUE;
}

/* Every window must belong to a class. We register ours here */
static BOOL NEAR RegisterWindowClass(hInstance)
HANDLE hInstance;
{

    PWNDCLASS pWndClass;
    HANDLE hTemp;

  /* Load the name string from resources */
    LoadString(hInstance, IDS_APPNAME,(LPSTR)szAppName,sizeof(szAppName));

  /* allocate space for the WNDCLASS structure and lock it down */
    hTemp = LocalAlloc(LPTR,sizeof(WNDCLASS));
    pWndClass = (PWNDCLASS)LocalLock(hTemp);

  /* fill the structure */
    pWndClass->hCursor  = LoadCursor(NULL, IDC_ARROW);  /* standard cursor */
    pWndClass->hIcon    = NULL;                         /* no icon */
    pWndClass->lpszMenuName = NULL;                     /* no class menu */
    pWndClass->lpszClassName = (LPSTR)szAppName;        /* our class name */
    pWndClass->hbrBackground = GetStockObject(WHITE_BRUSH); /*white background*/
    pWndClass->hInstance = hInstance;           /* instance handle */
    pWndClass->style = CS_VREDRAW | CS_HREDRAW; /* standard redraw values */
    pWndClass->lpfnWndProc = MainWndProc;       /* pointer to our window proc */

  /* register the class.  if fail, abort */
    if (!RegisterClass((LPWNDCLASS)pWndClass))
        return FALSE;

  /* free the memory used */
    LocalUnlock(hTemp);
    LocalFree(hTemp);

  /* show success */
    return TRUE;
}

/*
   If this not the first instance, we can retrieve static data from a
   previous invocation of the program
*/
static void NEAR GetPrevInstanceData(hInstance)
HANDLE hInstance;
{

    GetInstanceData(hInstance, (PSTR)szAppName, sizeof(szAppName));
    GetInstanceData(hInstance, (PSTR)szTimestr, sizeof(szTimestr));
    GetInstanceData(hInstance, (PSTR)szAM, sizeof(szAM));
    GetInstanceData(hInstance, (PSTR)szPM, sizeof(szPM));
```



```
    GetInstanceData(hInstance, (PSTR)&hFont, sizeof(HFONT));
}

/*
 Create the window, making sure that its position and size are suitable
 for the display.
*/
static BOOL NEAR MakeAndShowMainWnd(hInstance, hPrevInstance, cmdShow)
HANDLE hInstance;
HANDLE hPrevInstance;
int cmdShow;
{

    DWORD style;
    RECT clrect;
    HDC hIC;
    char szDisplayName[20];
    TEXTMETRIC TM;
    int displaywidth, displayheight;
    int width, height;

  /* First, create an information context for the display */
    LoadString(hInstance, IDS_DISPLAYNAME,
                          (LPSTR)szDisplayName,sizeof(szDisplayName));
    hIC = CreateIC((LPSTR)szDisplayName,(LPSTR)NULL, (LPSTR)NULL, (LPSTR)NULL);
  /* Now get the size of the system font */
    GetTextMetrics(hIC, (LPTEXTMETRIC)&TM);
  /* Also, retrieve information about the width and height of the display */
    displayheight = GetDeviceCaps(hIC, VERTRES);
    displaywidth = GetDeviceCaps(hIC, HORZRES);
  /* delete what we no longer need */
    DeleteDC(hIC);
  /*
     Now we specify the client window we would like to have.  However,
     its position is only nominal and later we will move it.  What we
     really want is the size of the window we need to create to achieve
     the desired client area.  Since the display is one line, the client
     must be high enough to display it.  And, since the text is an
     expression of the form hh:mm:ss A.(P.)M., it must be at least 13
     characters wide.
   */
    clrect.left = 0;
    clrect.top = 0;
    clrect.right = TM.tmAveCharWidth * 13;
    clrect.bottom = TM.tmHeight + TM.tmExternalLeading;

  /* this is the style we want */
    style = WS_OVERLAPPED | WS_MINIMIZEBOX | WS_CAPTION | WS_SYSMENU
                          | WS_THICKFRAME;
  /* adjust the rectangle to the actual window size */
    AdjustWindowRect((LPRECT)&clrect, style, FALSE);
  /* now compute the actual client width and height */
    width = clrect.right - clrect.left;
    height = clrect.bottom - clrect.top;
  /* create the window, letting it fall where Windows wants */
    hWndMain = CreateWindow((LPSTR)szAppName,
                            (LPSTR)szAppName,
                            style,
                            CW_USEDEFAULT,0,
                            width, height,
                            (HWND)NULL,
                            (HMENU)NULL,
                            (HANDLE)hInstance,
                            (LPSTR)NULL);

  /* if we fail, give up */
    if (hWndMain == NULL)
        return FALSE;
  /*
     if this is the first instance, we move the window
     to the lower right corner.
  */
    if (!hPrevInstance) {
      /* first, find out where we are */
        GetWindowRect(hWndMain, (LPRECT)&clrect);
      /* compute the width and height */
        width = clrect.right - clrect.left;
        height = clrect.bottom - clrect.top;
```

1988 Programmer's Journal 6.4

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Babcock & Hall, Continued

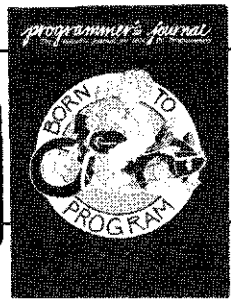
```
    /* now move the window down to the right corner */
       MoveWindow(hWndMain, displaywidth - width, displayheight - height,
                  width, height,FALSE);
       }
    /* finally, display the window and show success */
       ShowWindow(hWndMain, cmdShow);
       UpdateWindow(hWndMain);

       return TRUE;

    }
    /*
       Create a logical font that is suitable for displaying the time
       in our format when the window is iconic.  Since the font weight
       and size will depend on the display in use, three of the parameters
       can be read from WIN.INI.  For CGA type displays, try using a
       height of 6, a width of 4, and a weight of 400.  For the EGA,
       values of 8, 6, and 700 (the defaults) are satisfactory.
    */
    static HFONT NEAR GetFontInfo(hInstance)
    HANDLE hInstance;
    {
       char szKeystr[30];
       /* fill this structure with the font characteristics we want */
       LOGFONT clockfont;
       HFONT htemp;
       /* read the height from Win.ini */
       LoadString(hInstance,IDS_FONTHEIGHT,(LPSTR)szKeystr, sizeof(szKeystr));
       clockfont.lfHeight = GetProfileInt((LPSTR)szAppName, (LPSTR)szKeystr,8);

       /* read the width from Win.ini */
       LoadString(hInstance,IDS_FONTWIDTH,(LPSTR)szKeystr, sizeof(szKeystr));
       clockfont.lfWidth = GetProfileInt((LPSTR)szAppName, (LPSTR)szKeystr,6);

       /* read the weight from Win.ini */
       LoadString(hInstance,IDS_FONTWEIGHT,(LPSTR)szKeystr, sizeof(szKeystr));
       clockfont.lfWeight = GetProfileInt((LPSTR)szAppName, (LPSTR)szKeystr,700);

       /* default these values */
       clockfont.lfEscapement = 0;
       clockfont.lfOrientation = 0;
       clockfont.lfItalic = 0;
       clockfont.lfUnderline = 0;
       clockfont.lfStrikeOut = 0;
       clockfont.lfCharSet = ANSI_CHARSET;
       clockfont.lfOutPrecision = OUT_DEFAULT_PRECIS;
       clockfont.lfClipPrecision = CLIP_DEFAULT_PRECIS;
       clockfont.lfQuality = DEFAULT_QUALITY;
       clockfont.lfPitchAndFamily = FIXED_PITCH | FF_MODERN;
       clockfont.lfFaceName[0] = 0;

       /* get the handle to the font */

       return(CreateFontIndirect((LPLOGFONT)&clockfont));

    }
```

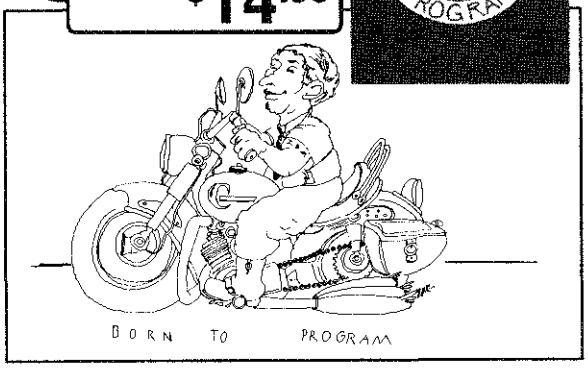
Listing 6

```
/*
   Resident segment for the Windows digital clock
*/

#include <windows.h>
#include <dos.h>
#include <string.h>

#define EXTERN          /* all global variables declared in this module */
#include "dc.h"

static struct dostime_t time;   /* holds the time parameters */
static RECT clrect;             /* formatting rectangle for time display */

/* local function declarations */
static void NEAR MainWndPaint(HWND, LPPAINTSTRUCT);
```


PJ ISSUE SPECIAL!

Born to Program T-Shirt & Back Issue Only $14.95



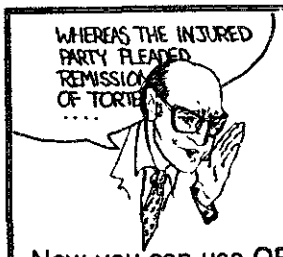


— PJ ISSUE SPECIAL —

BORN TO PROGRAM PACKAGE

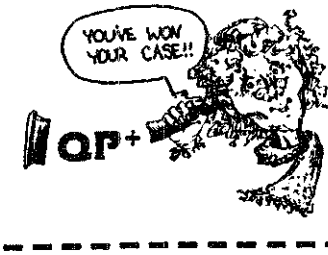
Back Issue 4.6— Born to Program
& Born to Program T-Shirt
Regular $19.00 Value—
For a Limited Time Only
Special Price $14.95

Born to program? Get your "colors" today!
See ad on page 76, offer expires September 1, 1988



Now you can use QPARSER+ to develop compilers, interpreters, complex user-interfaces, language & file format translators (i.e. Pascal to C, Bit Map to Postscript), language debuggers like lint, etc.

Develop language translators in C and Pascal within the IBM PC/XT/AT or VAX/VMS environments. A new user manual, automated syntax tree construction and an advanced code generation language are just a few of the improvements over the original QPARSER.



Another translation by QPARSER+™

Limited Time Offer Just $300 — FREE demo disk available

QCAD Systems
1164 Hyde Avenue, San Jose CA 95129 (408) 727-6884
Outside Calif. call TOLL-FREE (800) 538-9787

Number 189 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Multi-Edit vs. PIZZA

With EVERYTHING!

- Is your editor OUT TO LUNCH?
- Does it handle ALL OF YOUR NEEDS?
- Is it flexible, programmable and reconfigurable?
- MOST IMPORTANTLY, is it EASY TO USE?

OR WOULD YOU RATHER BE EATING PIZZA?

MULTI-EDIT DOS COMPLETE

VERSION 2

	Multi-Edit	BRIEF 2.0	Norton Editor	Vedit Plus	PIZZA WITH EVERYTHING
Edit 20+ Files larger than memory	Yes	Yes	No	Yes	12 slices
Powerful high level macro language	Yes	Yes	No	Yes	Italian
Full UNDO	Yes	Yes	No	No	No
Visual marking of blocks	Yes	Yes	Yes	No	Looks Good
Line, stream and column blocks	Yes	Yes	No	No	Use Knife
Automatic file save	Yes	Yes	No	No	No
Online help	Extensive	Limited	Limited	Limited	Extensive
Choice of keystroke commands or menu system	Yes	No	No	Yes	Menu Available
Function Key assignments labeled on screen (may be disabled)	Yes	No	No	No	No
Word processing functions	Extensive	Limited	Limited	Extra Cost	Difficult
Complete DOS shell	Yes	No	No	No	Deep Dish
Pop-up Programmer's Calculator and ASCII Table	Yes	No	No	No ASCII	No
Unlimited "Off the Cuff" keystroke macros	Yes	No	No	Yes	Sauce on Cuff often
Allocates all available memory to compiler when run from within editor	Yes	No	No	No	Lots of bytes
Intelligent indenting, template editing and brace/parenthesis/block matching and checking for C, PASCAL, BASIC and MODULA-2	Yes	C Only	No	Limited	Limited Intelligence
Flexible condensed mode display	Yes	No	Yes	No	Generally
PRICE	$99	$195	$50	$185	About $12

Get Our FULLY FUNCTIONAL DEMO Copy for only $10!

To Order, Call 24 hours a day:
1-800-221-9280 Ext. 951
In Arizona: 1-602-968-1945
Credit Card and COD orders accepted.

American Cybernetics
1228 N. Stadem Dr.
Tempe, AZ 85281

Requires IBM/PC/XT/AT/PS2 or full compatible, 256K RAM, PC/MS-DOS 2.0 or later. Multi-Edit and American Cybernetics are trademarks of American Cybernetics. BRIEF is a trademark of Underware, Inc. Norton Editor is a trademark of Peter Norton Computing, Inc. Vedit is a registered trademark of CompuView Products Inc. Copyright 1987 by American Cybernetics.

Only MULTI-EDIT tastes this good!

Fully automatic Windowing and Virtual Memory
Edit multiple files regardless of physical memory size
Easy cut-and-past between files
View different parts of the same file

Powerful, EASY-TO-READ high-level macro language
Standard language syntax
Full access to ALL Editor functions

Language-specific macros for C, PASCAL, BASIC and MODULA-2
Smart Indenting
Smart brace/parenthesis/block checking
Template editing
More languages on the way

Terrific word-processing features for all your documentation needs
Intelligent word-wrap
Automatic pagination
Full print formatting with justification, bold type, underlining and centering
Flexible line drawing
Even a table of contents generator

Compile within the editor
Automatically positions cursor at errors
Allocates all available memory to compiler

Complete DOS Shell.
Scrollable directory listing
Copy, Delete and Load multiple files with one command
Background file printing

Regular expression search and translate
Condensed Mode display, for easy viewing of your program structure
Pop-up FULL-FUNCTION Programmer's Calculator and ASCII chart

and MOST IMPORTANT,
the BEST USER-INTERFACE ON THE MARKET!
Extensive context-sensitive help
Choice of full menu system or logical function key layout
Function keys are always labeled on screen (no guessing required!)
Keyboard may be easily reconfigured and re-labeled
Extensive mouse support
Easy, automatic recording and playback of keystrokes
Anchovies easily removed

MULTI-EDIT COMBINES POWER WITH EASE OF USE LIKE NO OTHER EDITOR ON THE MARKET TODAY.

Number 184 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Babcock & Hall, Continued

```
/* Entry point for program */
int PASCAL WinMain(hInstance, hPrevInstance, lpszCmdLine, cmdShow)
HANDLE hInstance, hPrevInstance;
LPSTR lpszCmdLine;
int cmdShow;
{
    MSG msg;

    /* If initialization is not successful then exit */
    if (!InitProgram(hInstance,hPrevInstance, lpszCmdLine, cmdShow))
        return FALSE;

    /* Retrieve messages from Windows */
    while (GetMessage((LPMSG)&msg,NULL,0,0)) {
        TranslateMessage((LPMSG)&msg);
        DispatchMessage((LPMSG)&msg);
    }
    return msg.wParam;          /* exit program */
}

/* All messages are processed here */
long FAR PASCAL MainWndProc(hWnd,message,wParam,lParam)
HWND hWnd;
unsigned message;
WORD wParam;
LONG lParam;
{
    PAINTSTRUCT ps;

    static BYTE prevminute = 0;          /* these values control updating */
    static BYTE prevhour = 0;

    switch(message) {
      /* initialize the time structure when window is created */
      case WM_CREATE:
          dos_gettime(&time);
          break;

      /* we are exiting, so kill the timer and post a quit message */
      case WM_DESTROY:
          KillTimer(hWnd, ClockTimerEventID);
          PostQuitMessage(0);
          break;

      /* keep a record of the window size for use by the paint function */
      case WM_SIZE:
          clrect.right = LOWORD(lParam);
          clrect.bottom = HIWORD(lParam);
          break;

      /* Each timer event comes here for processing */
      case WM_TIMER:
          dos_gettime(&time);
        /* if new hour, erase window and redisplay the time */
          if (prevhour != time.hour)
              InvalidateRect(hWnd, (LPRECT)NULL, TRUE);
        /* otherwise, just redisplay the time without erasing background */
          else {
              if (IsIconic(hWnd)) {
                /* but, when iconic, do this only on a minute change */
                  if (prevminute != time.minute)
                      InvalidateRect(hWnd, (LPRECT)NULL, FALSE);
              }
              /* and if not iconic, display every second */
              else
                  InvalidateRect(hWnd, (LPRECT)NULL, FALSE);
          }
          prevminute = time.minute;    /* save the new values */
          prevhour = time.hour;
          break;

      /* we have to update the window */
      case WM_PAINT:
          BeginPaint(hWnd, (LPPAINTSTRUCT)&ps);
          MainWndPaint(hWnd, (LPPAINTSTRUCT)&ps);
          EndPaint(hWnd, (LPPAINTSTRUCT)&ps);
```


For those who need a

FAST C COMPILER with 8087 & software floating point and a full DOS 3 compatible STDIO library
FAST 8086 ASSEMBLER that is stand-alone, or inline with no separate assembly step
FAST LINKER that makes stand-alone .EXE files with both memory and disk-based overlays
SOURCE DEBUGGER for both C and Assembler
FULL SCREEN EDITOR that edits 2 files with no size limits & can invoke the Compiler & Linker
GRAPHICS package with line and text functions

DeSmet DC88

We've combined our popular DeSmet C88 Development Package with the DeSmet D88 Debugger and our Graphics Package. Library source code is available separately.

$99

C Ware
P.O. Box 428, Paso Robles, CA 93447
Phone:(805)239-4620 Telex: 358185 BBS: (805)239-4834
We accept VISA, MC & AMEX. Call now and we'll ship today
945 Spring St #14, Paso Robles, CA 93446

Number 38 on the Reader Service Card

C Math Toolbox©
Volume I
89
essential math functions
$89.00

A productivity tool for modern C programmers

Specifically designed for use with the Microsoft QuickC® and C Optimizing Compiler® 5.x under MS DOS®

Statistics
Complex numbers
Matrices, and more

For more information, contact your local dealer, or order by mail from:

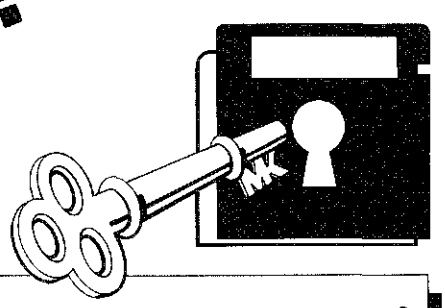
SILVER STATE SOFTWARE
1000 E. William Street, Suite 100
Carson City, NV 89701
800/624-0894 ext 874
Visa and MasterCard Welcome

QuickC, C Optimizing Compiler, and MS DOS are registered trademarks of the Microsoft Corp.

Number 186 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MASTER*KEY Unlocks Everything!



turn this into this!

```
C:\>DEBUG PROGRAM.COM
-D100 136
8648:0100  EB 18 49 6E 63 6F 72 72-65 63 74 20 44 4F 53 20  k.Incorrect DOS
8648:0110  76 65 72 73 69 6F 6E 0D-0A 24 50 B4 30 CD 21 86  version..$P40N!.
8648:0120  E0 3D 36 01 72 05 3D 0A-02 76 09 BA 02 01 B4 09  `=6.r.=..v.:..4.
8648:0130  CD 21 CD 20 58 EB 2F                             M!M Xk/
-Q
```

```
H00100: JMP     Short H0011A                    ;00100 EB18          __
;---------------------------------------------
        DB      "Incorrect DOS version"         ;00102 496E636F727265
        DB      0Dh                             ;00117
        DB      0Ah                             ;00118
        DB      "$"                             ;00119 24            $
;---------------------------------------------
H0011A: PUSH    AX                              ;0011A 50            P
        MOV     AH,30h                          ;0011B B430          _0
        INT     21h             ;1-DOS_Ver_Number ;0011D CD21        _!
        XCHG    AH,AL                           ;0011F 86E0          __
        CMP     AX,0136h                        ;00121 3D3601        =6_
        JB      H0012B          ; . . . . . .   ;00124 7205          r_
        CMP     AX,020Ah                        ;00126 3D0A02        =__
        JBE     H00134          ; . . . . . .   ;00129 7609          v_
H0012B: MOV     DX,0102h                        ;0012B BA0201        :__
        MOV     AH,09h                          ;0012E B409          __
        INT     21h             ;1-Display_String ;00130 CD21        _!
        INT     20h             ;TERM_normally:20h ;00132 CD20       _
;---------------------------------------------
H00134: POP     AX                              ;00134 58            X
        JMP     Short H00166                    ;00135 EB2F          _/
;---------------------------------------------


MASTER*KEY XREF - PROGRAM.XRF                                  Page 1

0102h                   :   121   2F5   301   320
020Ah                   :   126
03CBh                   :   12B
1-Display_String        :   130   591   610
1-DOS_Ver_Number        :   11D
H00100                  :   100
H0011A                  :   100   11A
H0012B                  :   124   12B
H00134                  :   129   134
H00166                  :   135
TERM_normally:20h       :   132
```

NOTE: The cross-reference is by memory location within the program file!

NOTE: The output is totally Microsoft MASM-compatible.

(not copy protected)

MASTER*KEY
No Other Product Comes Close!

An EXPERT may not know the solution, but always knows where to find it.

MASTER*KEY HELPS ANYONE solve those confusing and frustrating software puzzles more rapidly and easily than any other software available, at any cost! It gives you know-how within hours that may otherwise take years of experience. Create a new program from an old one. DON'T REINVENT THE WHEEL!

MASTER*KEY - Smart!
MASTER*KEY is an intelligent self-documenting MS-DOS reverse assembler. Its sophisticated procedures swiftly race through massive and baffling object code files to effortlessly discover potential trouble spots.

MASTER*KEY - Educational!
YOU DON'T NEED TO KNOW ASSEMBLY LANGUAGE! MASTER*KEY will take any program from your IBM-compatible computer and return fully-documented, easily-understood assembly language source code (Microsoft MASM 5.0 compatible).

MASTER*KEY - Easy To Use!
MASTER*KEY works both automatically from the DOS command line or interactively from menus similar to Lotus Corporation's 1-2-3 or Symphony. No need to remember any new commands or continually refer to a manual. Use it immediately!

Minimum System Requirements:
256K + 8088/8086/80186/80286/80386 PC
MS-DOS or PC-DOS 2.0 +
One 360K DSDD Floppy Drive (IBM PC Format)

MS-DOS is a trademark of Microsoft.
PC-DOS is a trademark of IBM.

MASTER*KEY will guide you step by step to:

1. Help you learn assembly language, if you desire.
2. Discover how any program runs or why it doesn't.
3. Alter or remove unwanted object code from any program.
4. Incorporate routines from compiled programs into other assembly language, Basic, C, or Pascal programs.
5. Make software more compatible with your computer. Be certain a questionable program won't damage your system BEFORE you run it.
6. Modify software to operate with other versions of DOS.
7. Customize your COMMAND.COM or other executable program directly or by reassembling your altered MASTER*KEY source code.

Order Now! Just $79.95

Phone orders accepted on MC or VISA
$82.45 (includes $2.50 shipping)
$87.65 in California (includes tax & shipping) **C.O.D.** orders add $2.00

(714) 596-0070

Dealer/Distributor Inquiries Welcome

Sharpe Systems Corporation
2320 E Street, Dept. 1610, La Verne, CA 91750 714-596-0070

MASTER*KEY should not be confused with any public domain or share ware software that may have a similar name or be a similar product.

Please send **MASTER*KEY!**

Send checks to:
Sharpe Systems Corporation
2320 E Street, Dept. 1610, La Verne, CA 91750

Name _____

Address _____

City _____ State _____ Zip _____

Number 175 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programmer's Tools

from The Software Family

Data Base

Btree/ISAM *by Softfocus*: The btree index structure provides rapid access to data based on content with multiple keys per file. Includes full C source code. ISAM automatically keeps data and indices synchronized, reducing your work compared to other btree packages. Multi-user functions available. For use with all popular C compilers. **$99**

QBase *by Crescent Software*: Qbase is a screen designer and relational database for BASIC that lets you quickly create applications including data entry, validation and file management. Qbase data validation is controlled by the database, so all input edits are automatically changed when you modify the database definition. Includes all source code. Specify QuickBasic or TurboBasic when ordering. **$79** ***Also available: Qbase Report $55***

Windows and Data Entry

Vitamin C *by Creative Programming* provides a comprehensive set of C functions for windows, menus, data entry & validation. Includes complete C source code. Specify your primary compiler when ordering. **$159**

VC Screen code generator for Viatamin C lets you paint your user interface to speed design, prototyping and maintenance. **$129**

386 Hardware

Hauppauge 386XT replaces your XT or compatible motherboard to provide true 80386 performance and maximum compatibility with XT/AT hardware, MS-DOS, OS/2 and Windows/386. 16Mhz, zero wait state. **$1195**

Hauppauge 386AT $1295

Free Catalog & Newsletter

The Software Family carries hundreds of products from dozens of publishers. Call or write for your free copy of our comprehensive catalog and newsletter.

Debugging and Quality Control

New version! **Periscope II** *by Periscope*: This is the debugger that lets you know where your program is getting hung up, even if you don't know where to set the breakpoint! Just press Periscope's hardware NMI button at any time to interrupt your program and enter Periscope. Periscope provides a comprehensive set of source level debugging capabilities including access to Microsoft C local variables and support for debugging Microsoft Windows. Don't waste hours guessing where to place you breakpoints or make do with less complete debuggers. For all popular languages and compilers. **$139**

Other Products

Borland **Turbo Pascal v4**	**$68**
Blaise **C Tools Plus/5.0**	**$95**
Digitalk **Smalltalk/286**	**$164**
Ithaca **HOOPS**	**$499**
Microsoft **C v5.1**	**$279**
Microsoft **Macro Assembler**	**$98**
Microsoft **OS/2 Programmer's Kit**	**$259**
Whitewater **ACTOR**	**$449**

The Software Family

800-443-7176

(415)957-0111

Visa • MasterCard
COD • Check

- Most orders shipped within 24 hours
- Free UPS ground shipping on orders over $100 ($3 on orders under $100. $2 COD fee)
- Federal Express, UPS air, and international service charges: our cost + $3 handling

The Software Family
649 Mission Street, Dept PJ078
San Francisco, CA 94105

Technical Support

The Software Family carefully evaluates features, quality and support before offering a product to our customers. All of the products featured in this ad are backed by excellent technical support from the publisher — in many cases directly from the program developers. All are published by companies with reasonable update policies and with a track record for responding to customer requests for both bug fixes and new features. We at The Software Family further back the products with our own technical support. We are willing and able to intercede with the publisher if your needs aren't satisfied or to research problems caused by the interaction of products we recommend. We can also provide quotes for custom modification of source-licensed products, in-depth technical consultation or custom-developed products.

Returns

The Software Family wants you to pay only for the tools you really use. If you order anything featured in this ad (or most of the products in our catalog) you can open and use the product for up to 30 days and still return it if it doesn't meet your needs.

Prices subject to change without notice - ©1988 TSF. All Rights Reserved.

Number 98 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

7

```
        break;

    /* all other messages come here */
    default:
        return ((long)DefWindowProc(hWnd,message,wParam,lParam));
        break;
    }
    return(0L);
}

/* Create a text string with the time and display it */
static void NEAR MainWndPaint(hWnd, lpps)
HWND hWnd;
LPPAINTSTRUCT lpps;
{
    char szTimeText[40];
    int i;
    BOOL iconic = IsIconic(hWnd);
    BYTE hour = time.hour;
    BYTE minute = time.minute;
    BYTE second = time.second;

    if (hour > 12)                  /* 12-hour display, i. e. 12:30 not 00:30 */
      hour -= 12;
    if (hour < 1)
      hour = 12;

    if (iconic) {
      strcpy(szTimeText, szTimestr);        /* Give icon display a title */
      i = strlen(szTimeText);
      szTimeText[i++] = '\n';               /* Force a new line */
      SelectObject(lpps->hdc, hFont);       /* Select in the small font */
    }
    else
      i = 0;

    if (hour > 9)                           /* ...hour...*/
      szTimeText[i++] = '1';
    szTimeText[i++] = hour % 10 + '0';
    szTimeText[i++] = ':';                  /* ...minute... */
    szTimeText[i++] = minute > 9 ? minute / 10 + '0' : '0';
    szTimeText[i++] = minute % 10 + '0';
    if (!iconic) {                          /* ...second...*/
      szTimeText[i++] = ':';
      szTimeText[i++] = second > 9 ? second / 10 + '0': '0';
      szTimeText[i++] = second % 10 + '0';
    }
    if (iconic)                             /* force another line if iconic */
      szTimeText[i++] = '\n';
    else                                    /* otherwise make a space */
      szTimeText[i++] = ' ';
    szTimeText[i] = '\0';

    if (time.hour > 11)                     /* morning or afternoon */
      strcat(szTimeText, szPM);
    else
      strcat(szTimeText, szAM);

    /* display the text */
    DrawText(lpps->hdc, (LPSTR)szTimeText, -1, (LPRECT)&clrect,
                        DT_NOCLIP | DT_EXTERNALLEADING | DT_CENTER);
}
```


Works with MS & TurboC

If you need the best source debuggers for MS-DOS or embedded microprocessor development with excellent support for C, PLM86, and assembly language contact:

SoftAdvances

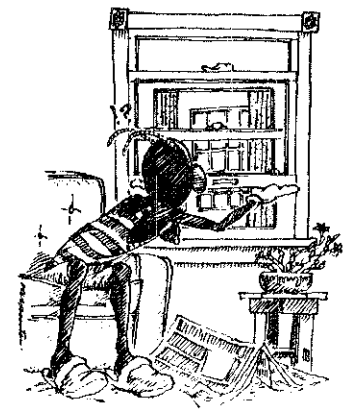
10811 Washington Bl., Ste. 205 • Culver City, CA 90232 • (213) 559-7015

Number 33 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Windows within Windows

by
John Navas II



Turbo C Version 1.0 quickly established Borland International as a major player in the C game. With Version 1.5, Borland upped the ante by including a video library that in some ways is even more powerful than the graphics library supplied with QuickC Version 1.0 from archrival Microsoft. The Turbo C video library includes not only graphic support, but also windowed text functions that can be used to create a powerful user interface for non-graphic applications. Let's take a close look at some of the innermost workings of these text functions.

The Turbo C library provides two different kinds of video output, controlled by the external variable **directvideo**. If **directvideo** is non-zero (the default), output will be written directly to the video screen; if **directvideo** is set to zero, output will be routed through the video BIOS. This gives the programmer the familiar choices of either speed (direct video) or compatibility (BIOS video).

Given that direct video is much faster than BIOS video and that direct video will work on virtually all IBM PC-compatibles, is there any point in using BIOS video? Yes. Direct video is considered "poorly-behaved" because it normally prevents software from performing properly in windowing, multitasking environments like TopView from IBM, the popular DESQview from Quarterdeck Office Systems, or Microsoft Windows. BIOS video solves the problem, but it has the drawback of further slowing down programs that already run more slowly in a multitasking environment.

Many programmers believe that these are the only two video alternatives. But there is a third alternative, one which provides fast text video that is "well-behaved" and fully compatible with both DOS and windowing environments. This third alternative is actually a hybrid of the other two since it uses special video BIOS functions to locate the memory to be used for direct video output. These video BIOS extensions were introduced by IBM as part of TopView, and both DESQview and Microsoft Windows support them as well. (All references to TopView below also apply to DESQview and Windows.)

TopView Video BIOS Extensions

Get Video Buffer is the key to the third alternative. **Get Video Buffer** must be called with the function number (FEh) in the AH register and the address of the real video screen (B000:0000h for a monochrome adapter; otherwise B800:0000h) in the ES:DI register pair; the address of the video buffer is returned in the same ES:DI register pair. If TopView windowing is active, a shadow buffer is assigned to each window and the address of that buffer is returned. The program then writes to the shadow buffer (which of course does not affect the real screen directly). If TopView windowing is not active, the buffer address is not changed, so the program will simply write to the real video screen automatically. [This is also an undocumented call supported by Microsoft Windows.—Ed.]

Update Video Buffer is used to inform TopView that part or all of the shadow buffer has changed. **Update Video Buffer** must be called with the function number (FFh) in the AH register, the address of the first character in the buffer that has been modified in the ES:DI register pair, and the count of consecutive characters that have been modified in the CX register. TopView is responsible for copying the appropriate characters (and attributes) from the shadow buffer to the real screen. If TopView windowing is not active, the function has no effect (and is not needed).

If no more than a few characters are to be changed, **Update Video Buffer** may be called once for each character; otherwise, it is more efficient to wait and make a single call for multiple changed characters—or even the entire shadow buffer.

Unlike TopView, DESQview automatically updates the real screen even if **Update Video Buffer** is not used. (However, the function call should always be included to maintain TopView compatibility.) On the other hand, even if **Update Video Buffer** is used, Windows won't update the real screen until your program relinquishes control. A good way to do that is to simply call DOS to check for keyboard input.

Note that the TopView extensions may be used only in text (not in graphic) modes.

Good News in Turbo C

Whew! Sounds pretty complex, doesn't it? If only Borland had seen fit to build this third alternative into the Turbo library—but no such luck. So the trick is to fool Turbo into using the TopView shadow buffer. Happily, once you know the secret, that turns out to be rather easy since the Turbo library still does most of the work.

The key is the undocumented **_VIDEO** structure in the Turbo C

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

library. (See the accompanying listing.) Whenever the screen is opened or the video mode is changed, Turbo stores all of the important video parameters in this structure, including the address of the video screen memory. If the address is changed—voila! Turbo will automatically write screen output to a buffer instead of to the actual screen.

The trick is to fool Turbo into using the windowing system's shadow buffer.

The function **SetupTopView()** takes care of changing the address (and shutting off the unnecessary CGA sync logic). **SetupTopView()** should be called at the start of a program and whenever the video mode is changed. Note that **SetupTopView()** also initializes the external char variable **TopViewActive** to non-zero if TopView windowing is active.

The other function, **UpdateTopView()**, is used to signal TopView to update the real screen from the shadow buffer. For simplicity's sake, **UpdateTopView()** specifies that the entire shadow buffer has changed. Used judiciously, this should not have a noticeable effect on performance.

So there you have it. Include these two simple functions in *all* your Turbo C programs, and you'll have "well-behaved" screen output (even windows within Windows) that is also very fast! ♦

John Navas II is a professional private computer consultant who specializes in high technology litigation. In the past seven years, John has worked for an international clientele that has included Intel and Hitachi.

The code from this article is available on 360K, MS-DOS disk, $10.00 postpaid. ORDER BY PHONE—VISA/MC orders call (503) 747-0800. Mail order information on page 76.

```
/*
**  FUNCTIONS THAT ALLOW TURBO C 1.5 VIDEO TEXT OUTPUT
**  ROUTINES TO RUN IN A TOPVIEW, DESQVIEW OR MICROSOFT
**  WINDOWS WINDOW, EVEN WHEN DIRECTVIDEO = 1.
*/

#include <conio.h>
#include <dos.h>

#define VIDEOINT    0x10    /* video BIOS interrupt */

extern struct _VIDEO {      /* UNDOCUMENTED VIDEO INFORMATION STRUCTURE */
    unsigned char winleft;      /* left window coordinate (0 relative) */
    unsigned char wintop;       /* top window coordinate (0 relative) */
    unsigned char winright;     /* right window coordinate (0 relative) */
    unsigned char winbottom;    /* bottom window coordinate (0 relative) */
    unsigned char attribute;    /* text attribute */
    unsigned char normattr;     /* normal attribute */
    unsigned char currmode;     /* current video text mode */
    unsigned char screenheight; /* bottom - top */
    unsigned char screenwidth;  /* right - left */
    unsigned char graphics;     /* mode type (0: text; 1: graphics) */
    unsigned char needcgasync;  /* cga sync to prevent "snow" (1: yes) */
    char far *videobuffer;      /* pointer to video buffer */
} _video;

char TopViewActive;         /* NON-ZERO IF TOPVIEW WINDOWING IS ACTIVE */

/*
**  SETUP FOR TOPVIEW/DESQVIEW VIDEO
**  call at startup and whenever the video mode is changed.
**  installs the shadow buffer address and sets TopViewActive.
*/
void SetupTopView(void)
{
    union REGS reg;
    struct SREGS sreg;

    if (_video.graphics) {                  /* if in a graphic mode */
       TopViewActive = 0;                  /* TopView cannot be active */
       return;
    }
    reg.h.ah = 0xFE;                        /* Get Video Buffer */
    sreg.es = FP_SEG(_video.videobuffer);   /* assumed buffer address */
    reg.x.di = FP_OFF(_video.videobuffer);
    int86x(VIDEOINT, &reg, &reg, &sreg);    /* call video BIOS */
    TopViewActive = (_video.videobuffer != MK_FP(sreg.es, reg.x.di));
    if (TopViewActive)                      /* if under TopView */
       _video.needcgasync = 0;              /* CGA sync is never needed */
    _video.videobuffer = MK_FP(sreg.es, reg.x.di);  /* shadow buffer addr */
}

/*
**  UPDATE SCREEN FROM TOPVIEW VIDEO BUFFER
**  (not required for DESQview)
*/
void UpdateTopView(void)
{
    union REGS reg;
    struct SREGS sreg;

    if (_video.graphics || !TopViewActive)  /* if graphic or not active */
    return;                                 /* TopView cannot be active */
    reg.h.ah = 0xFF;                        /* update video buffer */
    reg.x.cx = _video.screenheight * _video.screenwidth;    /* all of it */
    sreg.es = FP_SEG(_video.videobuffer);   /* buffer address */
    reg.x.di = FP_OFF(_video.videobuffer);
    int86x(VIDEOINT, &reg, &reg, &sreg);    /* call video BIOS */
    kbhit();                                /* yield Windows control */
}

/*
**  Test the TopView Video Functions.
*/

int main()
{
    int i;

    SetupTopView();
    cputs("This will run fine under DOS,"
       " as well as in a TopView/DESQview/Windows window!\n");
    UpdateTopView();
    for (i=1; i<=25; ++i) {
       sleep(1);
       cprintf("line %2d: %s\n", i,
           "ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz");
       UpdateTopView();
    }
    return 0;
}
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Proposed ANSI C Language Standard

by
Rex Jaeschke

© Rex Jaeschke, 1988. All rights reserved.

Rex checks Microsoft C 5.1 and QuickC against the (still) emerging ANSI Standard.

Microsoft C V5.1

On page 3 of the *Microsoft C Language Reference* manual you'll find the following note: "Microsoft is committed to conformance with the developing standard for the C language as set forth in the Draft Proposed American National Standard—Programming Language C."

Having just completed an extensive look at C V5.1, I believe them. Note that this article is not intended to be a review of the whole product—just of its ANSI C features or lack thereof. The issue of OS/2 support has no impact on this review.

Microsoft admits that there are minor differences between V5.1 and the version of QuickC shipped with it, and I have indeed found this to be the case. Except where noted below, all comments pertain to the optimizing compiler.

The Compiler Options

You can define the signedness of **char** objects on a global basis using the /J switch or on an individual basis using the **signed** and **unsigned** type modifiers.

The compiler recognizes the first 31 characters of identifier names, and there seems to be no way to change this except via the /H switch, which applies only to external names. The linker has a switch to handle case sensitivity of externals. For maximum portability, make the link case-insensitive and restrict external names to 6 or 8 characters.

For warning messages, you can choose an appropriate level—I always use /W3, but you have nowhere near the level of granularity you have in Turbo C, which permits you to trace a whole set of message classes on an individual class basis. The /Za switch flags extensions and other non-standard constructs.

One of the *very best* switches I have seen on any C compiler is /Zg. This switch constructs function prototypes from existing source files and writes them to **stdout**. By redirecting the output to a file, you have the basis for a header you can then **#include** in the appropriate places. This permits you to take full advantage of prototypes in your existing code at the expense of including the header at the start of each source file. This option alone is worth the price of the whole kit, and I wish other vendors would follow suit. A minor point about **stdio.h**, though: In this header, FILE is defined as a macro rather than via **typedef**. So when you pass a FILE argument to a function, the prototype generated by / Zg is

```
extern int f(struct _iobuf infile);
```

instead of

```
extern int f(FILE infile);
```

The Preprocessor

Microsoft C V5.1 has all of the new preprocessor capabilities including **#pragma** (which they use heavily); **#error**; the **#include** identifier; the **#**, **##**, and **defined** operators; **#elif**; and the predefined macros __DATE__, __TIME__, and __STDC__, as well as __FILE__ and __LINE__. In ANSI mode, __STDC__ will be 1, and Microsoft has chosen to define it with a value of 0 since they cannot yet claim to be ANSI-conforming—there is no standard yet.

I was still able to fool the preprocessor with some extreme tests, but it came out with very high marks. It permits benign macro redefinitions as required by ANSI; that is, it flags a macro if it is already defined differently and accepts the macro otherwise. Consider the following example:

```
/*1*/ #define MACRO1 a macro
/*2*/ #define MACRO1 a macro
/*3*/ #define MACRO1 a   macro    /* spaces */
/*4*/ #define MACRO1 a      macro /* tabs */
/*5*/ #define MACRO1 example
```

The compiler complained only about #5 since it does have a different definition. All the others are equivalent *provided* you ignore white space (which ANSI permits). However, ANSI does *not* require white space to be ignored, so another ANSI-conforming compiler may legitimately complain about #3 and #4 above, as well.

Miscellaneous ANSI Features

The following ANSI features are supported:

- full function prototypes and new style function definitions
- **signed** keyword
- **const** and **volatile** type modifiers (although **volatile** has no semantics)

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

- unary plus operator
- **void** functions
- concatenation of adjacent string literals
- the escape sequences **\xhhh**, **\a**, and **\v**
- the **long double** type
- **long double** and **float constants**
- generic pointers (**void ***)
- **%p** in **printf** for displaying pointers
- * mask in **printf** for variable field widths

The following ANSI features are not supported:

- trigraphs
- initialization of **auto** arrays and structures

The Runtime Library

All headers except **locale.h** (internationalization support) are provided. Except for those foreign language support functions recently added by ANSI, the V5.1 library seems to have the complete library, with the correct return types and argument lists. There are a few comments worth making, however:

- **assert** is implemented as a block statement, not as a **void** expression, and this prohibits its use in some situations.
- Surprisingly, **errno** is not declared in **errno.h**.
- Two macros are missing from **float.h**, and various other macros have been added (whose names are in the programmer's namespace).
- The integral minimum limits in **limits.h** are, unfortunately, not quite correct (at least in my opinion) even though they are ANSI-conforming. They reflect a ones-complement machine, and as such, CHAR_MIN is –127 instead of –128. Likewise for the other minima. I believe it is ANSI's intention that you specify your exact minima, not simply meet those specified by ANSI. I debated this with a Microsoft developer, and we had some disagreement.
- **stdarg.h** does not define the macro **offsetof**.
- **string.h** does not define the macro NULL.
- **math.h** has numerous macros whose names are in the programmer's namespace.
- **time.h** does not define the type **size_t**.

QuickC Notes

QuickC had the following limitations (not shared by the V5.1 compiler):

- can't handle macro definitions as large as the compiler can handle
- doesn't predefine __DATE__, __TIME__ and __STDC__
- permits __LINE__ and __FILE__ to be **#undefined**
- has no **#error**

In all other cases, it behaved as the compiler did.

Continued . . .

ANSI C Standard Meeting Report

Last week (April 18–22), I attended an ANSI C Standards Committee meeting in Nashua, New Hampshire. Late last year, it was our intent to vote the standard out at that meeting; however, that didn't happen.

At the December 1987, meeting in Austin, Texas, there was a last minute (last days, actually) push to incorporate some kind of syntax within a prototype to indicate to a compiler that the incoming arguments did not overlap. This was of particular interest to members implementing on systems with vector processing hardware. (Optimizers were also interested.)

As a result of this push, a palatable idea was proposed, and many of the more senior members of the committee worked into the wee hours over several days drafting precise language. The proposal was to add a new keyword, **noalias**, which would be a type specifier just like **const** and **volatile**. On the last day of the meeting, the **noalias** proposal was adopted, and we voted out the standard for its second public review—for a two month period. (Two members voted against this.)

Within days of that meeting's completion, word had reached the C community at large, and loud objections began to filter back to the committee. The most noticeable objection came from Dennis Ritchie, the principal designer of the C language. Not only did Dennis say he didn't like our solution, but he also submitted a public comment saying so and attended the Nashua meeting. (Interestingly, this was his first meeting.)

Dennis objected to the haste with which the keyword was added and didn't think it solved the problem. Dennis also thought it incorrect to use a type specifier as the solution. Dennis' final statement: "It must not survive."

Several others who had voted for **noalias** in December spoke out against it in April, and abetted by Dennis' persuasion, we voted **noalias**, and its associated baggage, out of the standard. The name **noalias** is now so tainted that it can never be used for anything again—it's cursed. Oh well, if you don't have bad, you can't have good.

All of this was achieved on the first day of the meeting. The rest of the week was dedicated to sifting through the several hundred pages of public comments. To expedite things, we broke into our usual 3–4 person subgroups. Each group had to screen their set of issues to identify those they thought needed full committee attention. While a few substantive changes were made to the draft, by far the largest number were editorial changes to both the draft and rationale documents.

On Friday morning, we again voted out the draft for another two month public review period. At the next meeting (August), we will repeat the same process as in Nashua (*sans* the **noalias** incident) and with a little luck could possibly vote out the final standard at that meeting. Of course, we could always choose (or be forced) to make one or more substantive changes, in which case, we go 'round again (and again, and again . . .)

In the meantime, I will be playing secretary at the next ISO C Standards meeting in London, UK, on June 13–14. P.J. Plauger, President of Whitesmiths, Ltd., is the convener. The meeting is to be held at the British Standards Institute.

To get a copy of the latest ANSI draft and rationale documents, contact Global Engineering Documents, Inc., by calling (800) 854-7179 or (714) 540-9870. Single copy price: US $65.00. Global Engineering are located in Santa Ana, California, USA, which is in the Pacific Standard Time (PST) zone.—RJ ♦

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DEVELOPMENT TOOLS FOR C

- A COLLECTION OF "TOOL BOXES" FOR THE DEVELOPMENT OF POWERFUL C PROGRAMS
- DESIGNED TO BE USED BY BOTH NEW AND EXPERIENCED C PROGRAMMERS
- EACH TOOL BOX IS FULLY DOCUMENTED -- WITH EXAMPLES

SYSTEM TOOL BOX
- ACCESS THE POWER OF YOUR PC
- KEYBOARD & SCREEN MANAGEMENT
- GRAPHICS & MUCH MORE
- 184 PAGE USER'S GUIDE

LIB + SRC $35

UTILITY TOOL BOX
- GENERAL PURPOSE FILTERS
- UTILITY PROGRAMS
- GENERAL PURPOSE FUNCTIONS
- 86 PAGE USER'S GUIDE

LIB + SRC $25

WINDOW TOOL BOX
- COMPLETE WINDOW CONTROL
- BUILD WINDOW DATABASES
- FULL ERROR REPORTING
- 84 PAGE USER'S GUIDE

LIB $45 -- SRC $125

MENU TOOL BOX
- VERTICAL/HORIZONTAL MENUS
- CREATE MENU DATABASES
- FULL ERROR REPORTING
- 78 PAGE USER'S GUIDE

LIB $35 -- SRC $75

- HARDWARE -- IBM PC, TI PRO OR TANDY 1000
- COMPILER -- MICROSOFT™ C (3.00 & LATER)

- CALL OR WRITE FOR FREE INFO & DEMO PROGRAM
- TECHNICAL SUPPORT IS UNLIMITED & FREE
- NO ROYALTIES

FLEMING SOFTWARE
5741 Wagonwheel Trail
Castle Rock, CO 80104
(303) 688 - 3584

Number 118 on the Reader Service Card

The Bottom Line

Microsoft V5.1 comes very close to the draft ANSI Standard of December 1987, and is very well positioned to cover the home stretch when a standard is finally voted out. In the two biggest areas of historical discrepancy, the preprocessor and library, Microsoft gets an "A."

Since I looked at Turbo C V1.0 several issues ago, I have developed an extensive evaluation suite that stresses the preprocessor and library. I plan to subject other compilers to this suite, including Turbo C V1.5.

Future ANSI C Meetings

The schedule for future ANSI meetings: August 15–19, Cupertino, California (hosted by Borland, HP, and Tandem); and December 12–16, Seattle, Washington (hosted by Microsoft). ♦

Rex Jaeschke is an independent computer consultant, writer, and lecturer. He is also a member of the X3J11 C Standard's Committee, observer on the ISO C Standard's Committee, Founding Editor of the C Journal, *and the C language Editor of* DEC PROFESSIONAL. *Rex's new book* Portability and the C Language *will be published by Hayden in September. It is the first text on this popular topic.*

If you would like to comment on or question any aspect of the Proposed ANSI C Standard, write to

Rex Jaeschke
2051 Swans Neck Way
Reston, VA 22091

Rex will endeavor to address your concerns in future columns as space permits.

"If you need source code, make sure your wallet is wide open or get VITAMIN C."
- Computer Language, June, 1987

Vitamin C

It's good for your system!

- Professional C function library
- 30 day money back guarantee
- Multiple bullet proof windows
- Easy full screen data entry
- Unlimited data validation
- Context sensitive help manager
- Menus like Lotus and Mac
- Programmable keyboard handler
- Text editor routines
- No royalties or runtime fees
- Library source included FREE
- Free technical support
- Free BBS at (214)418-0059
- Supports all major compilers including Microsoft 5.0
- VCScreen code generator too!
- UNIX version avaialable

Fast, flexible, versatile, reliable. Vitamin C delivers the vital combination that software professionals demand. Highly efficient, professionally crafted C code provides lightning fast displays required by today's window intensive programs.

High level functions provide maximum productivity and require little supporting code. Plus, Vitamin C's versatile, open ended design is full of hooks so you can intercept and plug-in special handlers to customize or add features to most routines.

VCScreen, our screen painter / code generator speeds your development even more! Simply draw your input forms using our interactive design editor and generate perfect C source code ready to compile and link with the Vitamin C library.

Vitamin C $225
Includes all source code FREE! For IBM PC, XT, AT, PS/2 and true compatibles. Specify compiler when ordering.

VCScreen $99.95
For IBM PC, XT, AT, PS/2 and true compatibles. Requires Vitamin C.

ORDER NOW!

We ship UPS. Please include $3 for ground, $6 for 2-day air, $20 for overnight, or $30 if outside the U.S. Texas residents add 7 1/4 % sales tax. All funds MUST be in U.S. dollars drawn on a U.S. bank. Visa & MasterCard accepted.

(214)416-6447

creative PROGRAMMING

Box 112097 • Carrollton, Texas 75011

Windows • Data Entry • Menus • Help • Text Editing
Plus... All Source Code FREE!

Contact this Advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

What have YOU been missing?

C Programmer's Toolbox Volumes I & II

CFlow™:	Determine program hierarchy, external R/T library functions, etc.;
CLint™:	More rigorously check program syntax;
CPrint™:	Beautify source programs to user selected formats;
CXref™:	Cross reference and check symbol usage;
CritPath™:	Determine a program's critical path;
PMon™:	Monitor program/OS execution; and more...

Create better, faster, higher quality and easier to read programs in a fraction of the time. Let your system do the work for you. With 23 powerful, state of the art tools for the IBM PC and compatibles, both beginners and experts will find programming a breeze.

Easy to use. Unlimited program sizes. Online documentation...

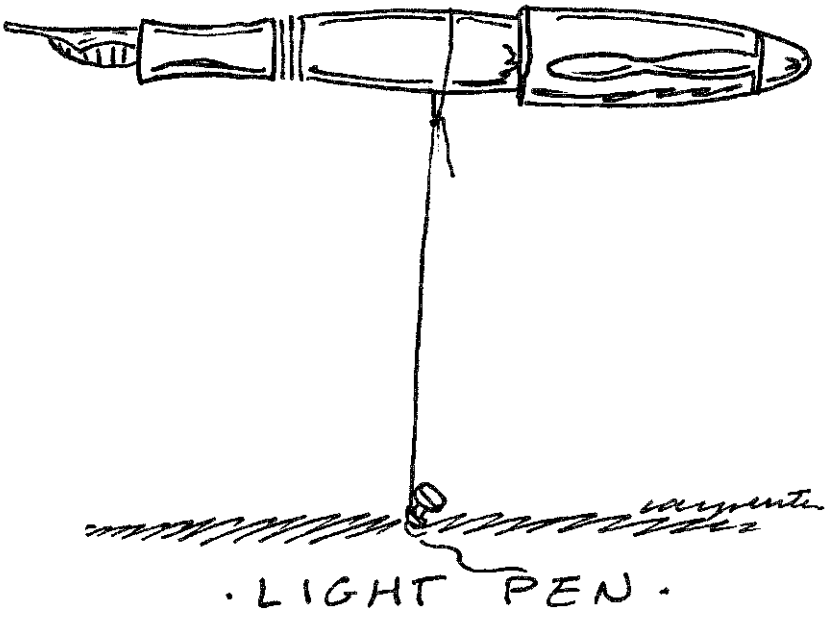
At $79.95 per volume or $130 for both with a 30 day money back guarantee, the Toolbox is simply the best value today. The Toolbox works with your existing C compiler(s) and enhances your development environment.

Call and Order Today
Visa, MasterCard Accepted

™

MMC AD Systems
Box 360845 Milpitas, California 95035
(408) 263-0781
"The C Tool Specialists"

Number 94 on the Reader Service Card



Locate C Bugs before they Bite with

PC-lint

PC-lint will analyze your C programs (one or many modules) and uncover glitches, bugs, quirks, and inconsistencies. It will catch subtle errors before they catch you. By examining multiple modules, PC-lint enjoys a perspective your compiler does not have.

"... a remarkable well thought-out product which will check for just about every conceivable coding error ... Its value increases with frequent use ... we confidently recommend PC-lint."

Andrew Binstock, The C Gazette

"PC-lint has everything going for it: flexibility, speed, good documentation, and a reasonable price. I exercised the product daily on a large, working, project to see if I could include it in my development tools. The answer is a definite YES."

Stephen D. Cooper, Blue Notes
San Francisco PC Users Group

NOW AVAILABLE
Shrouded Lint Source for use on VAX/VMS, Versados, OS/9, IBM VM/CMS - MVS, etc.
Requires only K & R C to compile.
Prices start at $798.
Call for details.

Gimpel Software
3207 Hogarth Lane
Collegeville PA 19426
(215)584-4261

PRICE: $139.00 first copy, $100 each additional, MC, VISA, COD, PA residents add 6% sales tax, Outside USA add $20.

Runs on MS-DOS -- K&R C with ANSI extensions -- direct support for 12 MS-DOS C compilers including MS 5.0, Turbo, Lattice, Desmet, Aztec

PC-lint is a trademark of Gimpel Software.

Contact this advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Not a Creature Was Stirring . . .

by
John J. Newlin

. . . well, not until Apple introduced the mouse as a point and shoot device with the Macintosh. Now there are lots and lots of mice stirring around PCs. In fact, it's currently estimated that ten percent of the installed base of over 10 million PCs are equipped with mice. While that may seem to be a low percentage (the Macintosh boasts 100 percent), the main reason it's not higher is the low percentage of software that actually performs better with a mouse (about 2 percent).

Some programs today are much easier to use and therefore more efficient with a mouse interface, and at least one, Microsoft Word, practically requires one. Most CAD/CAM, paint, and draw programs are clumsy and difficult to use without a mouse. Microsoft Windows and the programs designed to run in that environment are severely hampered without a mouse, and I venture to say that you would be hard pressed to find anyone running desktop publishing software who does so without a mouse.

In my view, no programmer planning to incorporate a menu style user interface into his or her software can afford to ignore the mouse—especially because incorporating mouse support code is relatively simple and straightforward. After I ported Arcmaster to Turbo Pascal 4.0 and implemented a pull-down, pop-up menu interface, a user commented that the program "cried for a mouse interface." I don't particularly like to use a mouse, but I have one, and I could see that he was right. There's no doubt in my mind that the popularity of the program increased when I released the version that supported a mouse.

A Standard Driver Interface

One factor that greatly simplifies programming a mouse interface is that there is a standard. Mouse Systems was the first company to market a mouse for PCs, and they were followed shortly by Microsoft. Fortunately for programmers, even though the resident drivers offered by the different mouse manufacturers differ, the interface to those drivers is standard. The programmer doesn't have to worry about users equipped with four or five different brands of mouse hardware and software. The interface is interrupt driven, and as we shall see, very easy to exploit in the form of organic mouse code.

The mouse operates with all PC text and graphics modes. Except for the manner in which it handles the cursor, the mouse will react the same way in all modes. To the mouse, the screen is an array of 600 pixels by 200 pixels, the same as in the high resolution mode of the Color Graphics Adaptor (CGA). The mouse driver scales the pixel position (in *x*/*y* coordinates) according to the video mode. When in the CGA high resolution mode (600 x 200), the mouse provides a pixel by pixel report of the cursor position. In the CGA medium resolution mode, it scales the *x* position (horizontal) report by multiplying by two and reporting only the even numbered *x* coordinates. In the CGA 80 column text mode and monochrome mode, both the *x* and *y* coordinate positions should be reported in 8 pixel increments.

Cursor Handling

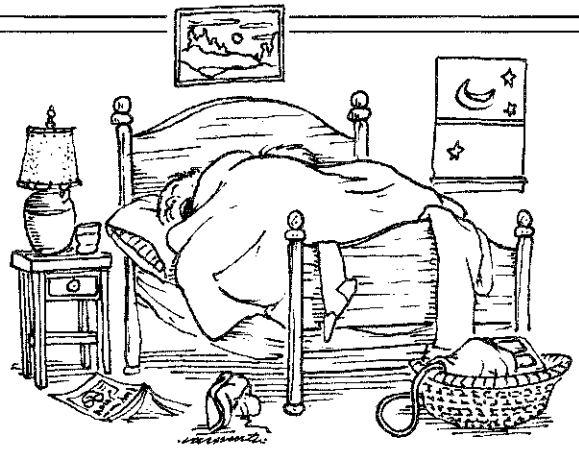
Cursor handling is a different story since the mouse driver supports three different types of cursors. The first of these is the normal hardware cursor controlled by the BIOS in the text mode. Control over the hard cursor is effected through calls to the video BIOS. The second cursor supported is the mouse soft cursor. The soft cursor is supported in the text mode, and the programmer can specify a cursor and a screen mask to produce a wide variety of cursor styles. The graphics cursor, the third cursor type supported, is available in the medium and high-resolution graphics modes. This cursor is controlled by a bit mask instead of a single character and therefore can be mapped accordingly into any shape that will fit into a 16 x 16 bit matrix. Both the soft and the graphics cursors can be turned on or off by the programmer through calls to the mouse interface. The PC hardware cursor can also be turned off by "hiding" it with a BIOS call.

Providing button release as well as button press information extends the programmer's control of the mouse environment.

Mouse Sensitivity

In this article, when I refer to *mouse sensitivity*, I'm talking about the ratio of mouse movement across the table (or pad) to the amount of corresponding cursor movement on the screen. As the mouse is physically moved, the mouse driver is sensing the mouse motion, transmitted either mechanically through action of the roller ball (Microsoft), optically (PC Mouse), or optomechanically (Logitech). The driver converts this motion into an incremental count proportional to the amount of movement sensed. Each unit of this count is equal to approximately .01 inch and is called, for reasons I cannot

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



fathom, a *mickey*. If the mouse is moved 1.50 inches to the left, the driver will count 150 horizontal mickeys. If the mouse is moved down .50 inch, the driver will count 50 mickeys.

The internal value of a mickey is always .01 inch and cannot be changed by the programmer, but the programmer can control the mickey-to-pixel ratio. The default values for the mickey-to-pixel ratio are 8 to 8 horizontally and 16 to 8 vertically. Some simple arithmetic here will show that with the default values, the mouse must be moved 0.08 inches horizontally to displace the cursor 1 text column and 0.16 inches vertically to displace the cursor 1 text row. Since the screen is a matrix of 80 columns and 25 rows, 6.4 inches of movement are required to traverse the screen horizontally and 4 inches to traverse it vertically. This means that, using default values, a user requires 25.60 square inches of desk area to access the entire screen. This can easily be changed under program control by setting the desired mickey-to-pixel ratio, which is variable from 0 to 32,767.

A Turbo Pascal Mouse Unit

Let's turn our attention now to the mouse driver. The driver is memory resident and is installed either as a device driver (MOUSE.SYS) or as an executable program (MOUSE.COM). All commercial mouse systems are delivered with a proprietary mouse driver that may operate through a serial port or through an installable bus card. The type of hardware interface and the architecture of the actual resident driver are of no concern to the programmer. The eighteen function calls to the driver are standardized and make it possible to include mouse service routines in a program with the assurance that the program will be properly responsive to any manufacturer's mouse. I'll be discussing each function call in detail in the following paragraphs, and each call will be implemented in a Turbo Pascal 4.0 routine. Access to the mouse driver interrupt will be via Turbo's INTR procedure. Register values will be set and retrieved using a variable declared as Turbo's intrinsic REGISTERS type. Personally, I eschew the Turbo mechanism whenever I can. I've coded a routine for the library of another Pascal compiler and can tell you that a lot of overhead is involved in saving and setting up the registers to make the call. If you want maximum speed and code compactness, use Turbo INLINE code instead. Those of you who view INLINE as tedious and difficult to write probably don't have Dave Baldwin's excellent public domain INLINE assembler (INLINE.ARC is available on CIS and many other BBS's).

Since the mouse routines will be coded in Turbo 4.0, I present them as a Turbo UNIT called MOUSE. At the beginning of Listing 1 you see the unit INTERFACE declarations for MOUSE unit.

Turbo 4.0 users will note that the implementation declarations have the parameter lists commented out in keeping with compiler requirements.

Initializing the Mouse

The first task of the MOUSE unit is to determine if a mouse driver is installed and to initialize it if it is. This is accomplished in the single Turbo function **InitializeMouse**. This function accomplishes four things. First, it determines if a mouse driver is present and responding through interrupt 33H. Second, it initializes a Boolean variable for later use that is set according to the availability of the driver. Third, it initializes all mouse settings to their default values, and fourth, it returns the number of buttons on the installed mouse. DOS Version 2 and some generic versions of DOS 3.x do not initialize the interrupt vector used by the mouse; therefore, you must test that interrupt 33h is not 0 before you invoke the mouse function vector. Otherwise, your program goes off the deep end and starts to execute whatever is located at 0000:00, hanging the machine. The mouse driver defaults are set by this function as follows:

Cursor	Screen center
Cursor status	Hidden
Soft cursor type	Inverse video box
Graphics cursor type	Arrow shape
Graphics cursor area	(-1,-1)
User call mask	Null
Light-pen emulation	Active
Horizontal mickey/pixel ratio	8/8
Vertical mickey/pixel ratio	16/8
Horizontal cursor limits	0 to 639
Vertical cursor limits	0 to 199

Buttons and Cursors

Now is a good time to talk about the number of buttons on the mouse. Microsoft has stubbornly (and wisely, I think) stuck with two while Logitech and Mouse Systems continue to provide three buttons. Extensive man-machine interface research has shown that more buttons are not necessarily better. A good example is the Macintosh mouse, which has only one button. I find that while the third button by itself can be useful, remembering what all the button combinations do can be a chore. In either case, this function will tell the programmer how many buttons are actually being serviced by the driver, and he or she can then decide how many buttons to support.

The next two routines control the visibility of the mouse soft cursor. They must be invoked alternately since a call to

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ShowSoftCursor actually causes a flag to be incremented and a call to **HideSoftCursor** decrements the flag. From the driver's aspect, if the flag is equal to -1, the cursor is to be hidden and if it is equal to 0, the cursor is to be shown. Two successive calls to either one of these routines will destroy the intended toggle effect. For this reason, I've added a cursor visibility variable (**CursorHidden**) that is toggled by each function and can be checked prior to calling either routine.

Mouse driver function 3 is quite useful and will probably be the most frequently invoked routine in this set. **MousePosition** returns a button status value that tells us which, if any, mouse button was pressed. It also returns the current horizontal and vertical position of the cursor. Which button has been pressed is a function of how the bits are set in the BX register upon return from call as follows:

Button(s) Pressed	Value of BX
None	0
Left	1
Right	2
Left & right	3
Middle	4
Left & middle	5
Right & middle	6
Right & middle & left	7

Notice that I've coded the call to function 3 to convert the returned raw horizontal and vertical pixel positions to coordinates scaled to the 80 x 25 text screen.

Mouse driver function 4 is the counterpart of function 3 in that it commands the driver to move the cursor to a specified position. If operating in the 80 column text mode, the horizontal and vertical cursor coordinates should be multiples of 8. Since the 80 column mode is the mode most likely to be in effect, I've coded this routine to convert from 80 x 25 screen coordinates to pixel values.

The next two driver functions (5 and 6) retrieve information on button presses and releases. Providing button release as well as button press information extends the programmer's control of the mouse environment. Both functions require an indication of which button is to be checked. Use the same values listed for function 3 for setting the BX register to indicate the desired button to check. Note that these functions also return the position of the cursor at the time of the last button press and that both Turbo routines scale the returned values for the 80 column text screen.

Mouse Driver functions 7 and 8 set the horizontal and vertical limits of the cursor. Once set, the cursor will be confined to the specified limits. Interestingly, it doesn't really matter which value, maximum or minimum, is placed in which register since the driver will automatically choose the lowest of the two as the minimum. For simplicity and economy of code, I have included setting both the horizontal and vertical limits in one Turbo procedure. Once again, the values are scaled so that the programmer can call the procedure with normal 80 column text values.

Graphics Cursor Shape and Type

We now come to the driver function that controls the shape of the graphics cursor and defines a cursor "hot spot" area. The hot spot is an area within the defined shape that determines which location will be reported at the time of a button press. This area is bounded by pixel coordinates that can vary from –16 to +16 for both axes, with the upper left corner defined as 0,0. You might notice that by using negative coordinate values, you can define a hot spot that is actually outside the cursor area.

The cursor shape is determined by the values contained in two consecutive arrays of 16 integers each. The first array is a screen mask that the driver will AND with a 16 by 16 bit screen area. The second array is a cursor mask that is XORed by the driver with the result of the AND operation to produce the final cursor. The address of the screen and cursor masks is placed in the ES:DX register pair. A simple but effective method to design the desired cursor shape is to use a 16 by 16 bit matrix as illustrated below:

```
    0 1 2 3 4 5 6 7 8 9 A B C D E F  Value
0                                    0000H
1                                    0000H
2                 1                  0080H
3               1 1 1                01C0H
4             1 1 1 1 1              03E0H
5           1 1 1 1 1 1 1            07F0H
6         1 1 1 1 1 1 1 1 1          0FF8H
7       1 1 1 1 1 1 1 1 1 1 1        3FFBH
8         1 1 1 1 1 1 1 1 1          0FF8H
9           1 1 1 1 1 1 1            07F0H
A             1 1 1 1 1              03E0H
B               1 1 1                01C0H
C                 1                  0080H
D                                    0000H
E                                    0000H
F                                    0000H
```

The values in the typed constant array **GraphicsCursorShapeMask** were derived from this matrix. The screen mask has all bits set to 1 so that the screen area under the cursor is visible. Note that **DefineGraphicsCursor** uses a Turbo untyped variable to pass the segment and offset address of the double integer array containing the screen and cursor mask bit values. Setting HotX and HotY each to a value of 8 will place the hot spot in the near center of the diamond cursor.

Mickeys, Lightpens, and User Defined Miscellaneous Calls

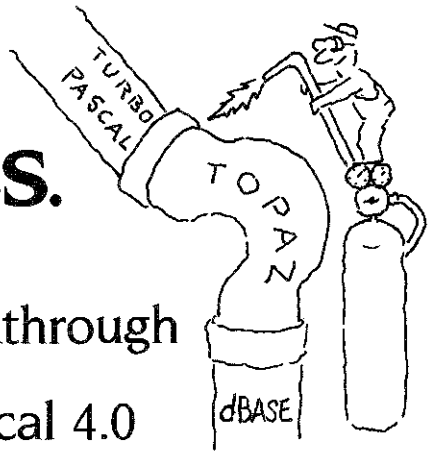
Function 11 of the mouse driver supports setting either the hard or soft text cursor. A value of 0 placed in the BX register selects the soft cursor while a value of 1 selects the hard cursor. The parameters **First** and **Last** define the start and end lines of the cursor, thus defining its shape. To completely hide the hard cursor on a CGA, set **First** to 9 and **Last** to 0.

Directly reading the mickey count (mouse motion counters) is accomplished through function 11. The mickey count can vary from 0 to 32,767, and direction is indicated by the sign of the returned values. A positive value is returned for motion to the right or downward, and a negative value indicates left or upward movement.

The mouse driver also provides the means to have a user defined routine called by the driver whenever a specified event occurs. The ES:DX register pair is set to the segment:offset address of a user defined procedure. The mouse driver makes a far call, and therefore the Turbo procedure should be marked accordingly using the **{$F+}** and **{$F-}** compiler directives. Your procedure must accomplish some housekeeping if this interface with the mouse driver is to be effected successfully. I suggest using the intrinsic Turbo 4.0 INTERRUPT directive, which will automatically save all the

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

YOU'LL LOVE THESE UTILITIES.



SAYWHAT?! The lightning-fast screen generator

TOPAZ. The breakthrough toolkit for Turbo Pascal 4.0

It doesn't matter which language you program in. With Saywhat, you can build beautiful, elaborate, colorful screens in minutes! That's right. Truly *fantastic* screens for menus, data entry, data display, and help-panels that can all be displayed with as little as one line of code in *any* language. Batch files, too.

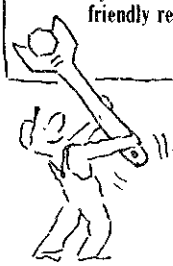
With Saywhat, what you see is *exactly* what you get. And response time is snappy and crisp, the way you like it. That means screens pop up instantly, whenever and wherever you want them.

Whether you're a novice programmer longing for simplicity, or a seasoned professional searching for higher productivity, you owe it to yourself to check out Saywhat. For starters, it will let you build your own elegant, moving-bar menus into any screen. (They work like magic in any application, with just one line of code!) You can also combine your screens into extremely powerful screen libraries. And Saywhat's remarkable VIDPOP utility gives all languages running under PC/MS-DOS, a whole new set of flexible screen handling commands. Languages like dBASE, Pascal, BASIC, C, Modula-2, FORTRAN, and COBOL. Saywhat works with all the dBASE compilers, too!

With Saywhat we also include a bunch of terrific utilities, sample screens, sample programs, and outstanding technical support, all at no extra cost. (Comprehensive manual included. Not copy protected. No licensing fee, fully guaranteed). **$49.95**

WE GUARANTEE IT!

IRON CLAD MONEY-BACK GUARANTEE.
If you aren't completely delighted with Saywhat or Topaz, return them within 30 days for a prompt, friendly refund.



If you'd like to combine the raw power and speed of Turbo Pascal with the simplicity and elegance of dBASE, Topaz is just what you're looking for. You see, Topaz (our brand new collection of units for Turbo Pascal 4.0) was specially created to let you enjoy the best of *both* worlds. The result? You can create truly dazzling applications in a very short time. And no wonder. Topaz is a comprehensive toolkit of dBASE-like commands and functions, designed to help you create outstanding, polished programs, fast. Think of it. With Topaz you can write Pascal code using SAYs and GETs, PICTURE and RANGE clauses, then SELECT and USE databases (real dBASE databases!), SKIP through records, APPEND data, and lots more.

In fact, we've emulated nearly one hundred *actual* dBASE *commands and functions*, and even added *new* commands and functions to enhance the dBASE syntax! All you have to do is declare Topaz's units in your source code and you're up and running!

The bottom line? Topaz makes writing sophisticated Pascal applications a snap. Data entry and data base applications come together with a minimum of code and they'll always be easy to read and maintain.

Topaz comes with a free code generator that automatically writes all the Pascal code you need to maintain a dBASE file with full-screen editing. Plus outstanding technical support, at no extra cost. (Comprehensive manual included. Not copy protected. No licensing fee, fully guaranteed). **$49.95**

ORDER NOW. YOU RISK NOTHING. Thousands of satisfied users have already ordered from us. Why not call toll-free, right now and put Saywhat and Topaz to the test yourself? They're fully guaranteed. You don't risk a penny.

SPECIAL LIMITED-TIME OFFER! Buy Saywhat?! and Topaz together for just $85 (plus $5 shipping & handling). That's a savings of almost $15.

To order: Call toll-free

800-468-9273

In California: **800-231-7849**
International: 415-571-5019

The Research Group
88 South Linden Ave.
South San Francisco, CA 94080

☑ **YES.** I want to try:

Saywhat?! your lightning-fast screen generator, so send _____ copies ($49.95 each, plus $5 shipping & handling) subject to your iron-clad money-back guarantee.

Topaz, your programmer's toolkit for Turbo Pascal 4.0, so send _____ copies ($49.95 each, plus $5 shipping & handling) subject to your iron-clad money-back guarantee

☐ **YES.** I want to take advantage of your special offer! Send me _____ copies of both Saywhat?! and Topaz at $85 per pair (plus $5 shipping & handling). That's a savings of almost $15.

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

☐ Check enclosed ☐ Ship C.O.D. ☐ Credit card

_____ Exp. date _____ Signature _____

THE RESEARCH GROUP

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

registers and set up the data and stack segments properly. However, since the Turbo INLINE procedure is designed to handle an interrupt call, it will exit using an IRET. Therefore, it will be necessary to include the following Turbo INLINE code so that your procedure exits with a return far (RETF):

```
Inline(
 $8B/
 $E5/
 $5D/
 $07/
 $1F/
 $5F/
 $5E/
 $5A/
 $59/
 $5B/
 $58/
$CB);
```

Function 12 is called with the ES:DX register pair pointing to your procedure and the CX register set according to which button press or release you desire as a trigger, as follows:

CX = 1	Cursor position change	(bit 0 set)
CX = 2	Left Button pressed	(bit 1 set)
CX = 4	Left Button released	(bit 2 set)
CX = 8	Right Button pressed	(bit 3 set)
CX = 16	Right Button released	(bit 4 set)

The user defined procedure must be of the form

```
Procedure MouseUserProc(AX,BX,CX,DX : integer);
```


TURBO SOURCE SEARCH

TSS is a programmer's SIG that currently has 240+ disks representing 80MB of archived files!! TSS specializes in locating publicly accessible files in Turbo Pascal, BASIC, QBASIC, C, QuickC, ASM and Modula-2. We offer these files by mail as well as on-line.

TSS-BBS (617-545-9131 within the PC-Pursuit area) is an outlet for the programming code from *Programmer's Journal*. All files in the TSS SIG are freely accessible through the BBS to members and non-members. Members have increased time and access. A central library for publicly accessible programming utilities is a valuable resource, and we are certain that programmers recognizing that will support our efforts by becoming members.

TSS is also a BBS-by-mail. All files maintained by TSS-BBS are available to members for as low as $2.50/disk. All files on all disks are archived to maximize the number of bytes per disk. A yearly membership to TSS, $40.00, entitles you to purchase disks from the library for $4.00/disk. Bulk discounts can lower the price to $2.50/disk. We also offer custom filled disks. Send $4.00 for the catalog disk or $3.00 for a printed catalog. (Members receive first catalog disk FREE.)

TURBO SOURCE SEARCH
P.O. Box 11, Scituate, MA 02066
Voice: 617-545-6677
Data: 617-545-9131, 1200/2400 BAUD, 24 hrs.
VISA/MC/Check/Money Order/non-US extra

Number **35** on the Reader Service Card

Upon return, the AX register contains the event mask with the corresponding bit set if the specified event occurred, the BX register contains the button status, and the CX and DX registers contain the horizontal and vertical mouse positions.

Light-Pen Support

Light-pen support is provided by driver functions 13 and 14, but since Turbo 4.0 lacks any built in light-pen support and there seems to be little interest in light-pen programming, I'll skip these functions.

Setting the Mickey-to-pixel Ratio

The sensitivity of the mouse can be controlled through function 15, which sets the mickey to pixel ratio. Sensitivity is determined by the number of mickeys (stored in the mouse driver motion counters) the mouse must be physically moved to effect a cursor movement of 8 pixels. The values of the parameters **Horiz** and **Vert** may vary from 0 to 32,767.

The last driver function is number 19 (what happened to 16,17, and 18, I don't know). This function sets the threshold for doubling the motion of the cursor. When physical movement of the mouse exceeds this threshold, the cursor will be moved twice as fast. After the threshold is exceeded, it must be turned off to return the cursor motion to normal. A method of doing this is to set the threshold to 32,767.

Listing 2 is a Turbo 4.0 program that uses the Mouse unit and demonstrates how you might integrate the mouse with the keyboard. This program doesn't actually do very much. It clears the screen, writes a five item menu, places the cursor bar over the first item, and then waits for either keyboard or mouse input. Mouse input can be in the form of a button press or mouse movement. If the mouse is moved up or down, the cursor bar is moved from item to item accordingly. The same movement will occur with the pressing of the up or down arrow keys. The "sensitivity" of the cursor bar can be adjusted by changing the values of the constants **HiLimit** and **LoLimit** in conjunction with the delay statement. Also note that if a mouse driver is not present, menu operation will still be normal and will be responsive to cursor pad keys.

The mouse can be a powerful user interface device, and providing organic support in your programs really won't cost you much in extra code. Remember that Turbo 4.0 will not link the procedures declared in the mouse unit that are not actually called by your program. If you're designing software that you expect a wide variety of people to use, many will be grateful for the opportunity to employ their beloved mice. ◆

John J. Newlin is a retired Naval Aviator who works for Hughes Aircraft Co. as a systems engineer. John holds degrees in mathematics and computer science and has programmed extensively in assembly, Turbo Pascal, Marshal Pascal, and Modula 2. John is the author of SCOUT, a memory-resident disk/file/directory manager, and ARCMASTER, a "point and shoot" style interface between the user and ARC and de-ARC programs. John is a charter member of the Association of Shareware Professionals. Readers may write to John at the following address:

John J. Newlin
4060-228 Rosenda Court
San Diego, CA 92122

Code follows.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available on 360K, MS-DOS disk, $10.00 postpaid. ORDER BY PHONE — VISA/MC orders call (503) 747-0800. Mail order information on page 76.

Code from "Not a Creature Was Stirring . . ." by John Newlin 1

Listing 1

```
UNIT MOUSE;

  INTERFACE

  Uses Dos;

  Const
    Mouse = $33;
    GraphicsCursorScreenMask : array[1..16] of word =
        ($FFFF,$FFFF,$FFFF,$FFFF,$FFFF,$FFFF,$FFFF,$FFFF,
         $FFFF,$FFFF,$FEFF,$FFFF,$FFFF,$FFFF,$FFFF,$FFFF);
    GraphicsCursorShapeMask : array[1..16] of word =
        ($0000,$0000,$0080,$01C0,$03E0,$07F0,$0FF8,$3FFB,
         $0FF8,$07F0,$03E0,$01C0,$0080,$0000,$0000,$0000);
    Buttons : integer = 0;

  Var
    Regs : Registers;
    MouseAvail,SoftCursorHidden : boolean;
    MouseButtons : integer;
    Function InitializeMouse : integer;
    Procedure ShowSoftCursor;
    Procedure HideSoftCursor;
    Function ReadMousePosition(var MouseX,MouseY : integer) : integer;
    Procedure SetMousePosition(MouseX,MouseY : integer);
    Function MouseButtonPresses(Button : integer; Var MouseX,MouseY : integer) :
      integer;
    Function MouseButtonReleases(Button : integer; Var MouseX,MouseY : integer) :
      integer;
    Procedure MouseCursorLimits(MinX,MaxX,MinY,MaxY : integer);
    Procedure DefineGraphicsCursor(HotX,HotY : integer; Var Mask);
    Procedure DefineTextCursor(HardOrSoft,First,Last : integer);
    Procedure MouseMickeyCount(Var MouseX,MouseY : integer);
    Procedure SetMouseUserProc(EventMask : integer; Var ProcAddr);
    Procedure MickeyToPixelRatio(Horiz,Vert : integer);
    Procedure HighSpeedThreshold(Threshold : integer);

IMPLEMENTATION

    Function InitializeMouse : integer;
    begin
      if  MEMW[$0000:$00CC]= 0 then
        begin
          InitializeMouse := 0;
          Exit;
        end;
      Regs.AX := 0;
      Intr(Mouse,Regs);
      MouseAvail := Regs.AX = 65535;
      if MouseAvail then InitializeMouse := Regs.BX else
        InitializeMouse := 0;
    end;

    Procedure ShowSoftCursor;
    begin
      Regs.AX := 1;
      Intr(Mouse,Regs);
      SoftCursorHidden := false;
    end;

    Procedure HideSoftCursor;
    begin
      Regs.AX := 2;
      Intr(Mouse,Regs);
      SoftCursorHidden := true;
    end;

    Function ReadMousePosition { (var MouseX,MouseY : integer) } : integer;
    begin
      Regs.AX := 3;
      Intr(Mouse,Regs);
      MouseX := succ(Regs.CX div 8);
      MouseY := succ(Regs.DX div 8);
      Intr(Mouse,Regs);
      ReadMousePosition := Regs.BX
    end;

    Procedure SetMousePosition { (MouseX,MouseY : integer) } ;
    begin
      Regs.AX := 4;
      Regs.CX := pred(MouseX * 8);
      Regs.DX := pred(MouseY * 8);
      Intr(Mouse,Regs);
    end;

    Function MouseButtonPresses { (Button : integer; Var MouseX,MouseY : integer) } :
                                integer;
    begin
      Regs.AX := 5;
      Regs.BX := Button;
```

Code continues . . .

TRUE
MULTITASKING
With
MultiDos Plus
"multitasking for the IBM-PC."

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in most languages.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
  - * Intertask message communication. Send/receive/check message present on 64 message queues.
  - * Task control by means of semaphores. Get/release/check semaphores.
  - * Change priority-256 priority levels.
  - * Suspend task for specified interval.
  - * Spawn and terminate external *and internal* tasks.
  - * Disable/enable multitasking.
  - * and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

Hardware/Software Requirements

IBM PC/XT/AT or true clone. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

only: $24.95 *OR* $99.95 with source code

Outside USA add $5.00 shipping and handling.
Visa and Mastercard orders only call toll-free: 1-800-872-4566, ext. 350., or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760
MA orders add 5% sales tax.

Number 109 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MISOSYS

SOFTWARE FOR YOUR P.C.

Order Now 800-MISOSYS (647-6797)

ED/ASM-86™

- 80X86 Assembly Language development

An integrated editor, assembler, linker, and debugger for 8086, 80186, 80286, and 8087 assembly and disassembly. All features are in a single EXE file.

- a full-featured line EDITOR with intra-line editing, block move and copy, partial save and load.
- a full-screen turbo-type editor is supported on CGA-equipped systems.
- the ASSEMBLER supports structured coding; can directly generate a COM or EXE file; assembly to memory for faster debugging.
- The DEBUGGER includes symbolic disassembly of arbitrary code to disk.

ED/ASM-86 [M-86-030] . . $124.95 + $5 S&H

RATFOR-86™

- Rational FORTRAN translator

A professional implementation of RATional FORtran by James Beard, Ph.D. Provides structure and greater portability to FORTRAN programs to reduce your programming time and effort dramatically over that required when writing directly in FORTRAN. Includes tutorial reference manual.

- compiler has extensive macro support
- "arith" macro for binary arithmetic
- short form READ and PRINT
- enhanced switch case support
- Requires a FORTRAN compiler

RATFOR-86 [M-86-73] . . . $99.95 + $5 S&H

MISOSYS Inc.
P.O. Box 239
Sterling, VA 22170-0239
703-450-4181 MC, VISA, CHOICE
800-MISOSYS/647-6797

VA residents add 4½% sales tax.
S&H: Canada add $1;
Foreign use S&H times 3

Number 121 on the Reader Service Card

Newlin Code, Continued


```
    Intr(Mouse,Regs);
    MouseX := succ(Regs.CX div 8);
    MouseY := succ(Regs.DX div 8);
    MouseButtonPresses := Regs.AX;
  end;

  Function MouseButtonReleases { (Button : integer; Var MouseX,MouseY : integer) } :
                                 integer;
  begin
    Regs.AX := 6;
    Regs.BX := Button;
    Intr(Mouse,Regs);
    MouseX := succ(Regs.CX div 8);
    MouseY := succ(Regs.DX div 8);
    MouseButtonReleases := Regs.AX;
  end;

  Procedure MouseCursorLimits { (MinX,MaxX,MinY,MaxY : integer) } ;
  begin
    Regs.AX := 7;
    Regs.CX := pred(MinX * 8);
    Regs.DX := pred(MaxX * 8);
    Intr(Mouse,Regs);
    Regs.AX := 8;
    Regs.CX := pred(MinY * 8);
    Regs.DX := pred(MaxY * 8);
    Intr(Mouse,Regs);
  end;

  Procedure DefineGraphicsCursor { (HotX,HotY : integer; Var Mask) } ;
  begin
    Regs.AX := 9;
    Regs.BX := HotX;
    Regs.CX := HotY;
    Regs.DX := ofs(mask);
    Regs.ES := seg(mask);
    Intr(Mouse,Regs);
  end;

  Procedure DefineTextCursor { (HardOrSoft,First,Last : integer) } ;
  begin
    Regs.AX := 10;
    Regs.BX := HardOrSoft;
    Regs.CX := First;
    Regs.DX := Last;
    Intr(Mouse,Regs);
  end;

  Procedure MouseMickeyCount { (Var MouseX,MouseY : integer) };
  begin
    Regs.AX := 11;
    Intr(Mouse,Regs);
    MouseX := Regs.CX;
    MouseY := Regs.DX;
  end;

  Procedure SetMouseUserProc { (EventMask : integer; Var ProcAddr) };
  begin
    Regs.AX := 12;
    Regs.CX := EventMask;
    Regs.DX := ofs(ProcAddr);
    Regs.ES := seg(ProcADDr);
    Intr(Mouse,Regs);
  end;

  Procedure MickeyToPixelRatio { (Horiz,Vert : integer) } ;
  begin
    Regs.AX := 15;
    Regs.CX := Horiz;
    Regs.DX := Vert;
    Intr(Mouse,Regs);
  end;

  Procedure HighSpeedThreshold { (Threshold : integer) } ;
  begin
    Regs.AX := 19;
    Regs.DX := Threshold;
    Intr(Mouse,Regs);
  end;

begin
  Buttons := InitializeMouse;
end.
```

Listing 2

```
Program MouseMenuDemo;

Uses Crt,Dos,Mouse;
Type
  Str30 = string[30];

Var
  Result : byte;
  i : integer;

Const
  MaxItems = 5;
  UpArrow = #200;          {Up arrow OR $80}
  DownArrow = #208;        {Down arrow OR $80}
  Home  = #199;            {Home OR $80}
  EndKey = #207;           {End OR $80}
  Inverse = $70;
  Normal = $0E;
```

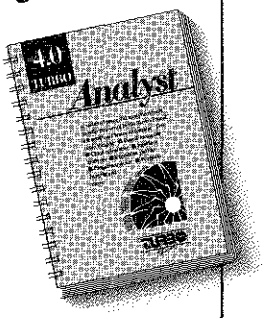
Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Write Better Turbo 4.0 Programs...

Or Your Money Back

You'll write better Turbo Pascal 4.0 programs easier and faster using the powerful analytical tools of **Turbo Analyst 4.0.** You get • Pascal Formatter • Cross Referencer • Program Indexer • Program Lister • Execution Profiler, and more. Includes complete source code.



Turbo Analyst 4.0 is the successor to the acclaimed TurboPower Utilities:

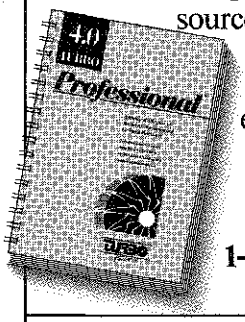
"If you own Turbo Pascal you should own the Turbo Power Programmers Utilities, that's all there is to it."

Bruce Webster, BYTE Magazine, Feb. 1986

Turbo Analyst 4.0 is only $75.

A Library of Essential Routines

Turbo Professional 4.0 is a library of more than 400 state-of-the-art routines optimized for Turbo Pascal 4.0. It includes complete source code, comprehensive documentation, and demo programs that are powerful and useful. Includes • TSR management • Menu, window, and data entry routines • BCD • Large arrays, and more.



Turbo Professional 4.0 is only $99.

Call toll-free for credit card orders.

1-800-538-8157 ext. 830 (1-800-672-3470 ext. 830 in CA)

Satisfaction guaranteed or your money back within 30 days.

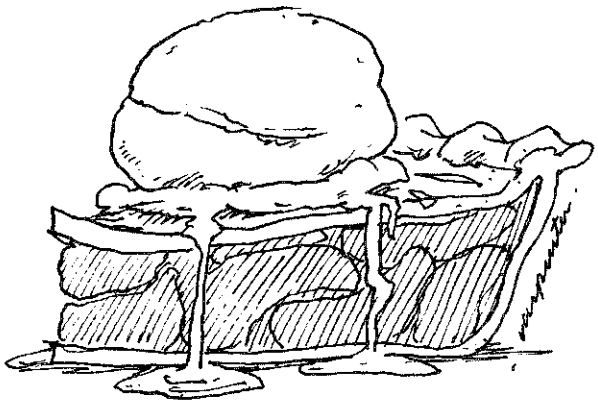
Fast Response Series:
■ T-DebugPLUS 4.0—Symbolic run-time debugger for Turbo 4.0, only $45. ($90 with source code)
■ Overlay Manager 4.0—Use overlays and chain in Turbo 4.0, only $45.
Call for upgrade information.

Turbo Pascal 4.0 is required.
Owners of TurboPower Utilities w/o source may upgrade for $40, w/source, $25. Include your serial number. For other information call 408-438-8608. Shipping & taxes prepaid in U.S. & Canada. Elsewhere add $12 per item.

TURBO Power

TurboPower Software
P. O. Box 66747
Scotts Valley, CA 95066-0747

Number 25 on the Reader Service Card



— REAL MODE —

NROFF/PC™

The REAL *Thing for* DOS

NROFF/PC is a complete text formatting system for *MS-DOS* systems. Including:

NROFF The *powerful* UNIX text formatter

TBL A tool to assist with the layout of tabular material in *Nroff* documents

MM A comprehensive *Nroff* macro package for preparing books and technical manuals

NEQN A tool for describing mathematical equations in *Nroff* documents

- All tools are a complete port from the AT&T Documentor's Workbench 2.0
- It's **Fast!** We've modified *Nroff* especially for DOS for lightning speed
- Supports any Dot Matrix printer and many laser printers
- Specially Priced At **$99**
- A complete *Troff* typesetting system is available **NOW** for LaserJet and PostScript printers on **MS-DOS** for *$695*, XENIX and Microport UNIX for *$795*.

Elan

Elan Computer Group, Inc.
410 Cambridge Ave., Suite A
Palo Alto, CA 94306
(415) 322-2450

Visa and MasterCard Accepted

Unix is a tradmark of AT&T
MS-DOS and Xenix are tradmarks of Microsoft

Number 173 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DOS & OS/2
Free Demo Diskette Available

Q. How many programmers does it take to maintain a MAKE dependency file?

A. NONE! *If you use* VersiMAKE™

VersiMAKE™ is a full-featured MAKE utility that includes:

- **Dependency Generation**
  Derives your system's dependencies, through analysis of its C and MASM source files. Say goodbye to manual maintenance of MAKE dependency files!
- **Wild Card File Name Matching**
  Analyzes an entire collection of source files with a single statement.
- **Nested Include Files**
  Handles standard C and MASM "include" conventions, and the INCLUDE environment variable.
- **Powerful Macro Facilities**
  Built-in macros, user-defined macros, and environment variables.
- **Analytical Reports**
  Shows the entire Include file hierarchy for each source file analyzed, and all of the parent source files for each Include file.

Q. How many programmers does it take to trace a symbol thru your system?

A. ONE! *If you use* VersiCREF™

VersiCREF™ is a unique utility that creates a Master Cross-Reference of your entire system.

- **Multi-Lingual**
  Handles C, assembler, or both.
- **Flexible**
  File names with line numbers, or file names alone. Global and local symbols, or globals alone.
- **Powerful**
  Easily handles systems containing 100 source files or more.

VersiMAKE™	$125
VersiCREF™	$75
Both	$150

(Outside U.S., add $5 for shipping and handling)

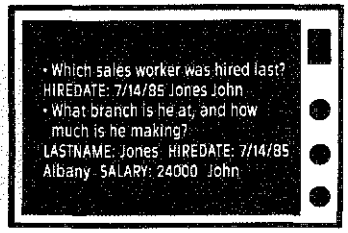
800-334-4096
(In NJ, 609-871-0202)
MC/VISA/AMEX accepted

SUMMIT INFORMATION SYSTEMS, INC.
73 East Lane, Willingboro, NJ 08046

Number 158 on the Reader Service Card


```
   LB  = 1;
   RB  = 2;
   LRB = 3;
   MB  = 4;
   LMB = 5;
   MRB = 6;
   LMR = 7;
   MenuItem : array[1..MaxItems] of str30 =
      (' Menu Item #1 ',
       ' Menu Item #2 ',
       ' Menu Item #3 ',
       ' Menu Item #4 ',
       ' Menu Item #5 ');

Function GetAKey : byte;
Var X1,Y1,x,y,Button,SaveAttr,Item : integer;
    KeyPress : boolean;
    Ch : char;
Const
    HiLimit = 16;
    LoLimit = 8;
Label Top;
begin
  x := 30;
  y := 9;
  Top:
  if MouseAvail then
    begin
      SetMousePosition(40,12);
      MickeyToPixelRatio(40,8);
    end;
    Button := 0;
    Repeat
      gotoxy(x,y);
      TextAttr := inverse;
      write(MenuItem[y-8]);
      KeyPress := Keypressed;
    If (not keypress) and MouseAvail then Button := ReadMousePosition(X1,Y1);
    Until Keypress or (Button  > 0) or (Y1  < LoLimit) or (Y1  > HiLimit);
    If Button > 0 then
     case Button of
       LB : Ch := chr(y-8);
       MB : Ch := Home;
       RB : Ch := #255;
     end
    else if (not KeyPress) and MouseAvail then
      begin
        if y1  > HiLimit then ch := DownArrow;
        if y1  < LoLimit then ch := UpArrow;
        delay(200);
      end
    else
      begin
       if button = 0 then Ch := ReadKey;
       if button = 0 then if Ch = #0 then Ch := Chr(Ord(ReadKey) or $80);
      end;
    TextAttr := Normal;
    gotoxy(x,y);
    write(MenuItem[y-8]);
    case ch of
      DownArrow : if y  < 13 then y := succ(y) else y := 9;
      UpArrow   : if Y  > 9 then y := pred(y) else y := 13;
      #13       : ch := chr(y-8);
      Home      : begin
                    y := 9;
                    goto top;
                  end;
      EndKey    : begin
                    y := 13;
                    goto top;
                  end;
    end;
    if ch in [UpArrow,DownArrow] then goto top;
    GetAKey := ord(ch);
  end;

Procedure Action(Item : byte);
begin
  clrscr;
  gotoxy(10,10);
  write('Menu item ',item,' was selected');
  repeat until keypressed;
end;

begin
  TextAttr := Normal;
  clrscr;
  DefineTextCursor(1,9,0);  {Hide the cursor on a CGA}
  for i := 1 to 5 do
    begin
      gotoxy(30,i+8);
      write(MenuItem[i]);
    end;
  repeat
    Result := GetAKey;
    gotoxy(1,25);
    if result in [1..5] then Action(result);
  until (Result in [1..5]) or (result = 27);
  DefineTextCursor(1,7,8);
end.
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED


• Which sales worker was hired last?
HIREDATE: 7/14/85 Jones John
• What branch is he at, and how much is he making?
LASTNAME: Jones HIREDATE: 7/14/85
Albany SALARY: 24000 John

announcing

ellie

version 2.0

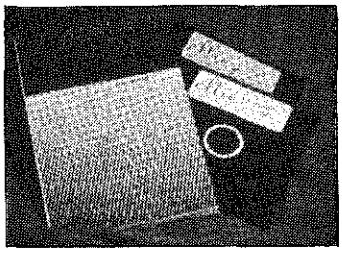
English in, dBASE out

. . . save hours of coding!

Try the amazing English-language interface for dBASE III/III +. (Now Fox Base + compatible!) 30-day money-back offer. Satisfaction guaranteed. Call 212-316-9078

"supports queries. . . that would otherwise require extensive JOIN and SET RELATION coding in dBASE." —*PCAI*, 1/88

"Using *ellie*, you can almost forget about dBASE commands." —*PCM*, February '88



only $99

ELFSOFT

210 W 101 NY NY 10025

Dealer inquiries invited.

Number **113** on the Reader Service Card

DEVELOPMENT TOOLS FOR GURUS, CONSULTANTS, VARS

SAVE TIME - MAKE MONEY

Our software SHELL simplifies coding & maintenance of PC systems that fill out forms. A de facto WORSTATION SPEC defines the SHELL structure.

- Programs delivered on four disks as X/SHELL:
  - Site licensed QBOS^ OP System
    - Hosted by CP/M or MS-DOS/UniDos
    - Interprets QBIC - a Basic dialect
  - Prototype MORTGAGE FORMS PROCESSOR.
    - Illustrates the QBOS^ USER MANUAL
    - Entree to a VERTICAL MARKET
    - Customize for other applications
      - Billing, shipping, etc.
  - Plain vanilla X/BOOK wordprocessor
  - Generic OP SPEC documentation:
    - 15 Pages in X/BOOK format
    - Customize for any Workstation

Call today for more information or
Send for $25 DEMO DISK & QBOS^ MANUAL:
Your check or we ship C.O.D.
Full refund if returned in 60 days.

X/ACT (714) 793-4915

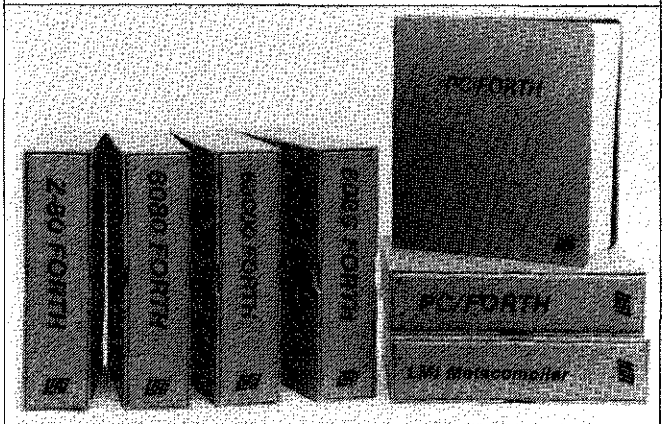
WORKSTATION SOFTWARE

JAVERT, INC.

P.O. Box 1144 Redlands, CA 92373 - (714) 793-4915

Number **202** on the Reader Service Card

TOTAL CONTROL with *LMI FORTH*™



For Programming Professionals:

an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

For Development: Interactive Forth-83 Interpreter/Compilers

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303
- No license fee or royalty for compiled applications

For Speed: CForth Application Compiler

- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

Support Services for registered users:

- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to: ***(213) 306-7412***

Overseas Distributors.
Germany: Forth-Systeme Angelika Flesch, Titisee-Neustadt, 7651-1665
UK: System Science Ltd., London, 01-248 0962
France: Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16
Japan: Southern Pacific Ltd., Yokohama, 045-314-9514
Australia: Wave-onic Associates, Wilson, W.A., (09) 451-2946

Number **63** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ ISSUE SPECIAL!

$19.00 value for only $14.95

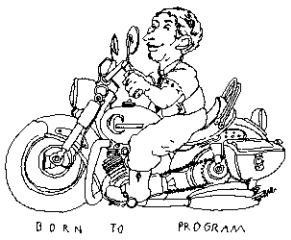
Born to Program T-Shirt
Back Issue 4.6
Offer expires September 1, 1988

T-SHIRTS

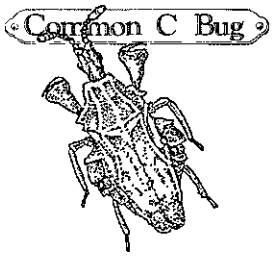
Multi-color imprints on the highest quality 50/50 cotton T-shirts.

$12.00 each — includes shipping & handling

Available in sizes **S, M, L, XL**



BORN TO PROGRAM



STACKUS PERNICIOUS

COMMON C BUG

Bugs Rule
New! 4 color imprint on a white shirt

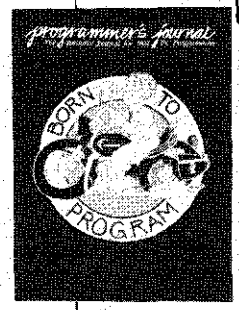
NEW

PARALLEL PORTS

PARALLEL PORTS

RS-232 head

RS-232 HEAD



BACK ISSUES

Back Issue 4.6
Nov./Dec. 1986

$7.00 each
Prepaid only

A few of the Back Issues currently available include:

Vol. 4.6: DTACK GROUNDED; Unix Books, Part 1; How many amoebas at one o'clock; DOS functions 45H and 46H; DOS Terminate ans Stay Resident; Turbo Pascal inline code; Pascal—Assembler souce code for DOS function calls; Alignment problems in C and Managing memory allocation; Proposed ANSI C Language Standard, Part 4; The Canon A1 laser printer; Genifer dBASE applications generator reviewed; Advanced Trace86 debugger reviewed.

Vol. 5.2: DTACK GROUNDED; Starting your own software house; Pink noise; TSR graphics print screen utility; Coding guidelines for resident programs; Parallel processing with the EGA; C—portability in practice and using header files; Proposed ANSI C Language Standard, Part 6; Indexing in BASIC; A first look at Turbo C.

Vol. 5.3: Starting your own software house--publicity; Unix books, Part 2; Consulting—merger mania; EGA data control; Modula 2 for Pascal lovers; Optimizing Turbo Pascal; ANSI C language standard; Coping with ANSI C and forcing file updates; Formatting BASIC output with buffered PRINT USING.

Vol. 5.4: Starting your own software house—technical support; Unix books, Part 3; Consultant productivity and effectiveness; EGA set/reset capabilities; ANSI C language standard; Character tables and string functions in C; MWWIND—a menu with windows utility; Variable passing and hiding in QuickBASIC 2 subprograms.

Vol. 5.5: Intel's 80386 vs. OS/2; Metagraphics President Jack Davis interview; Consulting—making computer tools pay for themselves; ASM—the invisible CapsLock; TSRs using expanded memory with Turbo Pascal; Note on optimizing Turbo Pascal; Proposed ANSI C Language Standard; Printer output from Turbo C programs; Matchup of Turbo Basic 1.0 and QuickBASIC 3.0.

Vol. 5.6: Multics, OS/2, paging and segmentation; The Usenix conference and other news; How to get rich in consulting; Shell in Marshal Pascal; Turbo Pascal 4.0 preview; Handling fixed length characater arrays in C; The proposed ANSI C Language Standard.

Vol. 6.1: Consulting is an attitude; Write mode 3 of the VGA; The LIM 4.0 expanded memory specification; Environment variables in Turbo Pascal 4.0; A timer program in C; Porting mainframe software to the PC--FORTRAN compilers under DOS and the Definicon 32-bit coprocessing engines.

Vol. 6.2: Hal Hardenbergh on RISCs, CISCs, and Fabs; Pink noise; Yet another VGA write mode; Inside Turbo Pascal 4.0; The proposed ANSI C Language Standard; Converting a Turbo C program to a TSR; 80286 and 80386 programming; 80386 software development; The Norton Guide for BIOS/DOS Assembly.

Vol. 6.3: Hal Hardenbergh on memory trends; Pink noise; Getting your consulting tools in order; Gordon Letwin on OS/2 memory management; Virtualizing the 386; Pascal—A CASE of the jumps; Turbo Professional 4.0; BASIC—CALL, GOSUB, and multi-line functions

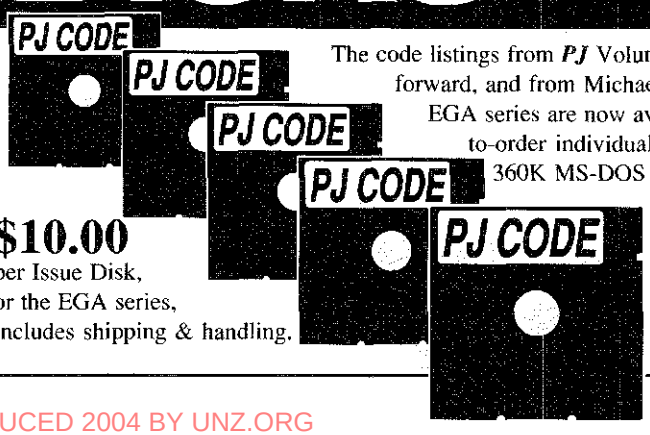
10% discount on any order of 10 or more T-Shirts, Disks or Back Issues.

These are just a few of the ***PJ*** back issues currently available— For a complete listing of all the back issues, check the box on the reader service card and send it in , or call us and ask for the ***PJ*** **Back Issue Index.**

CODE DISKS

ALL ORDERS MUST BE PREPAID.

Use the order form on the card opposite. Or save time— order by phone using your VISA/MC— CALL: (503) 747-0800



The code listings from ***PJ*** Volume 5.5 forward, and from Michael Abrash's EGA series are now available on easy-to-order individual issue diskettes. 360K MS-DOS disks standard.*

$10.00
per Issue Disk,
or the EGA series,
includes shipping & handling.

* Other formats available

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Windowing and Graphics

Microsoft Mouse Programmer's Reference
Microsoft Corporation
P.O. Box 97017
Bellevue, WA 98073
(800) 426-9400

The *Microsoft Mouse Programmer's Reference Guide* ($25) comes with utility programs to design mouse menus, two object library files to build mouse support into applications, and programming examples (one 5.25-inch disk). The first four chapters describe how to use the Mouse Menu software to design menus. The last four chapters detail the mouse software interface, mouse functions, making function calls from various languages, and writing mouse programs for IBM EGA modes. The VGA chip is not mentioned. Examples are provided to show how to interface with the BASIC interpreter, QuickBASIC 2.0, Microsoft Pascal, FORTRAN, C, and MASM. A special EGA Register Interface library is provided for writing applications for EGA graphic modes D, E, F, and 10H. This library saves the state of various write-only EGA registers so that the mouse cursor will be updated correctly. Brief appendices cover mouse driver command-line switches, differences between Versions 5.0 and 6.0 mouse drivers, making calls to Borland Turbo Pascal, and using a Hercules card.

GEM/3 Programmer's Toolkit
Digital Research, Inc.
P.O. Box DRI
Monterey, CA 93942
(800) 443-4200

Digital Research (DRI) has released the latest versions of GEM (Graphics Environment Manager)—the GEM/3 Desktop ($49.95) along with a revised GEM/3 Programmer's Toolkit ($500). The GEM programming environment provides two sets of functions—AES (Application Environment Service) and VDI (Virtual Device Interface)—for controlling graphics device output and pointing device and keyboard input. The GEM Programmer's Toolkit includes C-language bindings to facilitate calling the functions and to help develop portable programs. The kit also provides a GEM resource construction program, a series of C-language include files, and a symbolic debugger. GEM/3 supports more graphics devices than GEM 2.2 and adds more font support and greater speed of operation (redraw speed on CGA and VGA cards is up to four times faster). GEM/3 AES allows the choice of either drop-down or pull-down menus and control of menu bar activation. The GEM/3 VDI allows a simpler program invocation scheme (similar to that currently used by Xerox's Ventura Publisher 1.1 under GEM). GEM/3 also provides greater support for runtime selection of output destination, page size and orientation, number of copies, justified text, and rotated and scaled bit images, and more fonts are supported. The DOS version comes bundled with Bitstream's Fontware package, which allows generating screen and printer fonts in any size from 6 to 72 points (HP Laserjet II). The package comes with both Helvetica and Times Roman style fonts in normal, bold, italic, and bold italic along with Bitstream Charter—a newer Roman style font.

Toolkit documentation (500+ pages) includes three volumes: (1) *GEM AES* (higher level window and object functions); (2) *GEM VDI* (lower level graphics primitives); and (3) *RCS*—Resource Construction Set (utilities for constructing graphic resources and the debugger). An addendum documents changes and additions from GEM 2.2 to GEM/3. The list of GEM/3 compatible devices is extensive, with graphics displays (120+) ranging from low resolution monochrome and color to ultra high resolution, pointing devices (12 different mice and PC keyboard), tablets, scanners, camera devices, and about 80 output devices—plotters, phototypesetters, monochrome, color, and laser printers.

Prospero GEM Pascal & FORTRAN
Prospero Software
100 Commercial St., Suite 306
Portland, ME 04101
(207) 874-0382

Prospero Software from England is one example of the recent trend to export to the U.S. market better European software. For some time, now, Prospero has sold DOS Pascal and FORTRAN compilers into the U.S., and last fall the company opened a U.S. office to provide better marketing and technical support for their products. Closely following their increased U.S. presence, Prospero has released new Pascal and FORTRAN compilers designed specifically for GEM, for both DOS (80x86) and the Atari ST (68000). In the European market, DRI's GEM and Concurrent-DOS have consistently held a larger market share than in the U.S. And in Britain, the Atari ST is a more popular personal computer than it is here.

The Prospero GEM Compilers (Pascal at $149 and FORTRAN at $199) include a programming environment with a workbench to control the editor, compiler, linker, librarian, cross-reference utility, and symbolic debugger using pull-down menus and a multi-window display. The workbench editor supports the standard WordStar control key sequences along with cursor and special function keys and user-defined macros. All four compilers have full GEM VDI and AES bindings. In order to use the MS-DOS compilers, you also need to buy a version of the GEM Desktop. Each compiler comes with a three volume set of documentation (800+ pages) describing (1) the specific compiler, (2) the language VDI bindings, and (3) the language AES bindings. Both the DOS Pascal and the FORTRAN compilers have been validated for compliance by the National Bureau of Standards. The versions for MS-DOS (8086) are similar to Prospero's regular MS-DOS compilers except that they lack support for the 8087 coprocessor and only the large memory model is supported. Also, the GEM DOS linker does not support overlays. The GEM VDI and AES Pascal and FORTRAN bindings are especially convenient since DRI's GEM toolkit is based on C and assembly language bindings. If you are developing software targeted at both the Atari ST and the IBM PC (for example, music software), these compilers might provide you a portable window to both worlds.

Standards

System V Interface Definition
AT&T Customer Information
Attn: Customer Service Representative
P.O. Box 19901
Indianapolis, IN 46219
(800) 432-6600

Several years ago, in order to help standardize the programmer's interface to Unix and promote portability, AT&T drafted and published the System V Interface Definition (SVID). The SVID Release 2 consists of three attractive perfect bound manuals documenting Unix System V. These publications are intended to serve as a single reference source for the definition of the external interfaces to services provided by all System V environments, to define all additional services (such as networking and data management) provided as extensions, and to serve as a complete definition of System V interfaces for application source code.

Volume I (320 pages) covers the base system definition—the basic Unix Library and Operating System (OS) calls. Volume I also covers Kernel Extensions to support process accounting, software development (profiling and debugging), and enhanced interprocess communication (messages, semaphores, and shared memory). Volume II (460 pages) covers the Base Unix utilities, Advanced utilities, and Administered utilities—the commands available under System V Unix. Volume II also details software development utilities (the C Preprocessor and Compiler, the linker and debugger, lex, yacc, make, etc.) and the terminal interface definition (Curses and Terminfo databases). Volume III (380 pages) contains addenda to the Base System Definition and documents new capabilities introduced to System V with Release 3.0.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Volume III also covers changes to Terminfo, Curses, and network extensions—the Streams I/O, Shared Resource Environment, and Open Systems Network (OSI Model) interfaces.

I found the three volume set an excellent reference to Unix commands, libraries, and the Terminfo/Curses terminal interface, and also to how the equivalent C library routines and kernel services might be expected to function under DOS C compilers. If you are porting programs from Unix to DOS, these volumes could help you write Unix compatible DOS library functions not supported by your DOS C development environment. Single volumes ordered separately cost $37, while the entire set is priced at $84.

C Math Libraries

C Scientific Library
Eigenware Technologies
13090 La Vista Drive
Saratoga, CA 95070
(408) 867-1184

The C Scientific Library (CSL) from Eigenware Technologies is a scientific programming library of over 400 math functions. CSL is supplied as seven libraries available for Microsoft C 3.0 through 5.x and Borland's Turbo C. CSL costs $275 for the entire 400+ function library and is also available as three separate libraries (math, statistics, and matrix) for $99 each. CSL functions are provided as linkable object libraries compiled for the small, medium, and large memory models using the 8087 if present or the 8087 emulator (/FPc compiler option). Source code in C for CSL is available for an additional $75. The CSL functions were originally developed and tested using the VAX11 C and Computer Innovations' v2.3 C compiler and were later converted for Microsoft C. The C source code should work with other C compilers, and batch files for compiling and creating linkable libraries are included. A series of example programs is provided.

The CSL function library is comprehensive, covering matrix computation and decompositions, eigensystems and singular value analysis, matrix inversions, matrix multiplexing, linear algebra, differential equations of any order, quadrature, complex variables, polynomials, interpolation, table lookup, simple statistics, probability, analysis of variance, regression, time series, data analysis, discrete and fast Fourier transforms, nonlinear equations, optimization, finance, calendar, sorting, data plotting, and graphics support.

Documentation (over 600 pages) consists of three volumes: (1) *General Information and Tutorial*; (2) *Reference Manual* (the big one); and (3) *Graphics and Plotting*. The documentation is wide ranging, but needs work. The reference manual has no page numbers, the table of contents shows section heads only, and none of the manuals has an index. The reference manual is organized into eleven sections and contains a separate entry for each CSL function describing its purpose, a function header with argument list, a functional description (the algorithm used), function usage (input arguments and output), a code fragment example, references (the source of the algorithm), and related CSL functions. References are noted by book source only, not by pages. The Tutorial makes some interesting comments about a few algorithms, but in general algorithms are not discussed. Appendices to Volume I contain math constants and C function prototypes (also in supplied C header files) and a cross reference of CSL functions that notes other CSL functions and the C library routines required. CSL appears to be a complete package whose documentation needs some improvement to make it easier to use.

C Math Toolbox—Volume I
Silver State Software
1000 E. Williams St., Suite 100
Carson City, NV 89701
(702) 885-8922

The C Math Toolbox (Volume I for $89) from Silver State provides 89 math functions designed specifically for use with Microsoft QuickC and C 5.x DOS compilers. The functions in the Toolbox are delivered in three separate forms: (1) C source code; (2) optimized linkable Microsoft C library; and (3) CodeView linkable library. The source code files are in the proper form to be used with either Microsoft 5.x or QuickC. The linkable

OMNIVIEW
ROCK SOLID MULTITASKING!

Packed with the power you need, OMNIVIEW takes you beyond the limits of DOS.

- Time-slices multiple jobs
- Takes full advantage of EMS 4.0
- Optional support for '386 Systems
- Manages megabytes of programs
- Controls resident utilities
- Lets you cut and paste between programs

OMNIVIEW unleashes your PC & converts it into an easy-to-use, multitasking workstation. Whether you have an 80386 or 8088 based computer, OMNIVIEW will squeeze out every ounce of power.

Solid & bulletproof, it's 47K assembly language engine gives you all the power without added overhead or complexity. Just ask the reviewers.*

"Sunny Hill provides true multitasking for most PC applications. It performs remarkably well, causing almost no noticeable reduction in the performance of each program . . . The user manual is well organzied and easy to read . . . It is an excellent product."
PC Tech Journal Sept '87

"It does an amazing job of handling not only multitasking, but in many cases, concurrent processing as well. I was amazed at it's success in handling the "impossible" combinations I threw at it."
"It is one of the best designed & implemented programs I've ever worked with. It is also one of the most bulletproof."
"What sets it apart from the pack is the fact that it runs in a network environment."
Data Based Advisor Aug '87

OMNIVIEW requires an IBM PC, XT, AT or Jr compatible, and PC or MS DOS 2.x to 3.x. To get your copy, call toll free:
(800) 367-0651
or send $89.95 + 3.00 S&H ($8.00 Intl.) to:

Sunny Hill Software
PO Box 55278, Seattle, WA 98155-5278
For more information call (206) 367-0650 *Omniview formerly Taskview
DoubleDOS trademark Softlogic Solutions, Concurrent PC-DOS reg. trademark Digital Research Corp., MS Windows trademark Microsoft Corp., DESQview trademark Quarterdeck Office Systems, Inc. TopView trademark IBM Corp.

Number 85 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

libraries are compiled using the Medium Memory Model to conform to the default QuickC model using inline 8087 code. A set of MAK files (QuickC makefiles) for using the library with QuickC is also provided. The C source code functions should work with other C compilers, and two batch files are provided to recompile and build the libraries. The batch files may be modified for use with other compilers and to generate other memory models. For each math function, an example test program in C is also provided.

The math functions provided cover the following areas: byte matrix manipulation, complex math, real and complex matrix math, solutions to equations, fast Fourier transform, polynomial interpolation, function roots, simple integration, polynomial manipulations, simple statistics, regression, and time series analysis. Documentation consists of a single perfect bound manual of about 270 pages with a table of contents and an index. In general, the source code is not commented and the user is referred to the manual. The manual contains a separate section on each math function: its purpose, other functions used, a C prototype, the function arguments, notes for QuickC use, a reference for the algorithm, and an illustrative example program. The documentation is readable and well organized. The actual algorithms are not explained. References are noted only by book source, not by pages. The references seem a bit dated; none is post 1978, and most are from a decade earlier (this is only Volume I, though, with more to follow). Although the code seems to be translated rather directly from FORTRAN, most optimizing compilers don't really care and will generate the same code for i=i+1 as for i++. But some C coding improvements for speed could probably be made. This appears to be a solid, easy to use package if the areas covered fit your needs.

Compilers and Languages

NDP C-386 Compiler

MicroWay
P.O. Box 79
Kingston, MA 02364
(617) 746-7341

MicroWay has released Version 1.4 of their 80386 C Compiler ($595). Their NDP FORTRAN-386 compiler was released about 6 months ago (see *PJ* 6.1). Now C-386 is off the press, and Pascal-386 is to follow soon. Working with Green Hills Software (Glendale, CA), MicroWay has been porting their family of Unix compilers to DOS for the 80386 using the Phar Lap DOS Extender. MicroWay NDP C-386 generates optimized native 80386 code, supports 80287, 80387 and Weitek math coprocessors with inline code, and includes a DOS compatible C runtime library. The runtime library also includes graphics routines written in 386 assembly language, trace and debugging aids, and support for installing your own interrupt handlers.

Green Hills makes advanced optimizing compilers for FORTRAN 77, C, and Pascal that run on Unix System V, Berkeley Unix, and now 386/DOS. These compilers optimize globally, providing some of the fastest code execution in the business. NDP C-386 is a full implementation of PCC (Bell Labs' Portable C Compiler). The compiler generators optimize 80386 assembly language as output that is then assembled and linked using either Phar Lap 386 development tools on DOS or VAX machines or Unix V utilities. Programs may be run on DOS using Phar Lap's 386 DOS Extender. C-386 generates programs with no limit on size up to the 16 or 24 megabyte limit of most 80386 DOS machines (DOS Extender) or the 4 gigabyte limit under Unix V. A large number of command-line switches are available to control optimizations, extensions, and coprocessor support. Documentation consists of a thorough (340+ pages) manual covering the compiler and its use, C language elements and extensions (some ANSI C draft standard and Microsoft features are supported), interfacing to other MicroWay languages, numeric coprocessor support, trace/debug features, and the runtime library routines. In addition to the compiler, you must buy the Phar Lap Development tools (MicroWay also sells these).

WATCOM C6.0

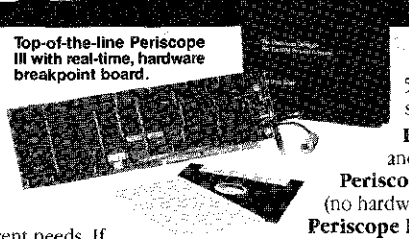
Watcom
1430 Massachusetts Ave, Suite 306
Cambridge, MA 02138
(800) 265-4555

Watcom has released their WATCOM C6.0 compiler for 80x86 DOS machines. Watcom, a Canadian software house, is not exactly new to the compiler market. They have developed language compilers for machines ranging from the IBM 370 to 8-bit microprocessors. Traditionally, Watcom C compilers have been available only on IBM 370 mainframes. WATCOM C6.0 is a high performance optimizing C compiler and development system (introductory U.S. price $295). The development system includes their ANSI compatible optimizing compiler, an editor, a full screen source-level debugger, runtime libraries, an overlay linker, an object librarian, a make utility, an object module disassembler, and EXPRESS C (Watcom's integrated fast C development system). The compiler supports five memory models (small, medium, compact, large, and huge) under DOS and can generate 8086, 80186, or 80286 code. The 80x87 math chip is supported with both inline code generation and an emulator library.

WATCOM C has the capability to generate the fastest code under DOS of any C compiler on the market. It achieves this feat by squeezing the last bit of speed out of

PERISCOPE™ POWER

...Keeps you going full steam ahead when other debuggers let you down. With four models to pick from, you'll find a Periscope that has just the power you need.



Top-of-the-line Periscope III with real-time, hardware breakpoint board.

. . . Start with the model that fits your current needs. If you need more horsepower, upgrade for the difference in price plus $10! And don't worry about having a lot more to learn . . . Even when you move to the most powerful model, Periscope III, an extra dozen commands are all that's involved.

Periscope's software is solid, comprehensive, and flexible. It helps you debug just about any kind of program you can write . . . thoroughly and efficiently. Periscope's hardware adds the power to solve the really tough debugging problems.

Periscope requires an IBM PC, XT, AT, or close compatible (Periscope III requires hardware as well as software compatibility); DOS 2.0 or later; 64K available memory; one disk drive; an 80-column monitor.

Periscope I includes a half-length board with 56K of write-protected RAM; break-out switch; software and manual for $345.

Periscope II includes break-out switch; software and manual for $175.

Periscope II-X includes software and manual (no hardware) for $145.

Periscope III includes a full-length board with 64K of write-protected RAM, hardware breakpoints and real-time trace buffer; break-out switch; software and manual. Periscope III for machines running up to 8 MHz is $995; for machines running up to 10 MHz, $1095.

Call Toll-Free for free information or to order your Periscope today!

MAJOR CREDIT CARDS ACCEPTED. **800-722-7006**

The PERISCOPE Company, Inc.
1197 PEACHTREE ST.
PLAZA LEVEL
ATLANTA, GA 30361
404/875-8080

Number 6 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

the 8086. The compiler does the conventional compiler optimizations, but the ultimate performance is a result of parameter passing conventions. Depending on their sizes, arguments are passed either in registers or on the stack. Registers AX, DX, BX, and CX are used first (in that order) to pass arguments to functions before the stack is used. The registers may be grouped together and are used to pass char, int, long, float, double, and both near and far pointers. The 80x87 registers are also used to pass floating point parameters to functions. Since many C functions may have only a few parameters, all the arguments may be passed in registers without being pushed onto the stack. These features may be controlled with #pragma directives on an individual function basis. A side effect of this optimization is that runtime and utility libraries have to be rewritten (if assembly language) or recompiled with WATCOM C to benefit from this enhancement.

The compiler supports a variety of command-line switches to control optimization level, memory models, cpu and math support, warning level, stack checks, debugging, language extensions, and other features. A convenient option allows outputting function prototypes to a file. Language extensions include support for cdecl, pascal, fortran, near, and far function modifiers. Supported pragmas allow powerful extensions and controls including inline assembly language, parameter passing in registers, forcing arguments into specific registers, exporting symbol names, using 80x87 registers (for paramater passing, return function values, and preserving registers across calls), returning structure pointers in registers, and calling near or far functions with FORTRAN or Pascal conventions. EXPRESS C supports the medium memory model and allows for quick development. The exhaustive documentation is spread across five manuals—(1)*Optimizing Compiler and Tool User's Guide*, (2)*Editor User's Guide*, (3)*Library Reference*, (4)*Language Reference*, and (5)*Express C User's Guide*. In short, WATCOM C6.0 is a powerful package for program development.

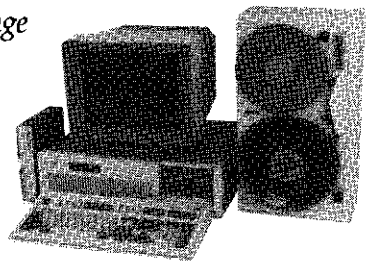
Guidelines C++ Version 1.2
Guidelines
P.O. Box 749
Orinda, CA 94563
(415) 254-9393 or 254-9183

Need some class for your next programming project? Object oriented programming has been gaining popularity, and C++ provides extensions to the C language to support objects and other levels of abstraction. Guidelines has ported Version 1.2 of AT&T's C++ translator to the DOS environment. Guidelines C++ ($295) is a full implementation and requires Microsoft C 3.0, 4.0, or 5.x and 640K of RAM. Guidelines provides a C++ preprocessor (CPRE.exe) and C++ translator (CFRONT.exe) to generate standard C code compatible with Microsoft C. This translated C code is then compiled and linked with the regular Microsoft C runtime libraries and Guidelines' C++ runtime library. Five memory models are supported—small, compact, medium, large, and huge. Batch files are provided to automate the process. The CPRE preprocessor supports command-line switches to define or undefine names and add search paths for include files. CPRE implements most of the ANSI draft standard. The CFRONT program supports several command-line options to control warning and diagnostic messages, add debug information, use a list of object sizes and alignments from an external file, accept regular (non-ANSI) C function declarations, and generate line number information. The C++ translator was developed in the Unix environment, where memory size is not a consideration, but it's possible to run out of memory in the DOS version (CFRONT is a 285K executable file). If you look at the actual C output files from the translator, you'll see that they are a bit difficult to interpret, with #line directives and no C pretty formatting. But these line directives let the C compiler report the errors back to the C++ input file.

Documentation consists of *The C++ Programming Language* by Bjarne Stroustrup, the current equivalent of Kernighan and Ritchie for C++, and a Guidelines C++ trans-

9-Track Tape Subsystem

The Data Interchange Solution You've Been Waiting For!



Qualstar's ½" 9-track Ministreamers™ bring full ANSI data interchange capability to your PC or Macintosh™ computer system.

With 9-track tape, you are free to exchange data files with any mainframe or minicomputer in the world.

Our affordable Ministreamers come in both 7" and 10½" versions and use less desk space than an ordinary sheet of paper. They can provide 1600 thru 6250 BPI capability and may be used for disk backup as well as data interchange. Complete subsystem prices, including software, start as low as $2,495 for 7" units and at $3,670 for 10½" units.

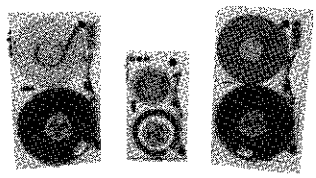
Qualstar has become the market leader in desk-top 9-track systems for a good reason; our tape drives have established an outstanding record for reliability and low cost of ownership.

Discover the many advantages 9-track tape has over other Micro/Mainframe links.

Call us today!

QUALSTAR®

9621 Irondale Avenue,
Chatsworth, CA 91311
Telephone: (818) 882-5822



Macintosh is a trademark of Apple Computer, Inc.

Number 97 on the Reader Service Card

C Scientific Library — CSL

Programming library for computations in research and mathematical analysis, on desk-top computers. Each routine is designed and documented for use by technical specialists and programmers.

ADVANCED MATRIX COMPUTATION and DECOMPOSITIONS
EIGENSYSTEMS and SINGULAR VALUE ANALYSIS
MATRIX INVERSIONS, MULTIPLEXING, and LINEAR ALGEBRA
DIFFERERENTIAL EQUATIONS OF ANY ORDER
MULTI-DIMENSION QUADRATURE
COMPLEX VARIABLES and POLYNOMIALS
INTERPOLATION and MULTI-DIM TABLE LOOK-UP
STATISTICS, PROBABILITY and ANALYSIS OF VARIANCE
LEAST SQUARES, REGRESSION and TIME SERIES
DISCRETE and FAST FOURIER TRANSFORMS
NONLINEAR EQUATIONS and MULTI-DIM OPTIMIZATION
FINANCE and CALENDAR FUNCTIONS
SPECIAL MATH FUNCTIONS including FITTING and EVALUATION
GRAPHICS and DATA PLOTTING

- Available for either Microsoft or Turbo C compilers — $279
- Tutorial and Examples — Over 400 functions and Three volume manual
- Source code available, optional — Add $70

Personal CSL — $85

- Includes $60 credit when ordering the complete CSL package with source code
- Math toolbox derived from C Scientific Library — 100 Functions
- 100 Examples — 100+ Manual pages

EIGENWARE TECHNOLOGIES 13000 La Vista Drive Saratoga, CA 95070
(408) 867-1184

CA residents add 6.5% sales tax. P.O. or Personal check

Number 29 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DISK FORMAT CONVERSION

XenoCopy-PC

PC-DOS program lets your PC

$79.95 + $5.00 S/H
Sales Tax if CA.
VISA MasterCard UPS COD

READ / WRITE /
FORMAT / DUPLICATE
Disks from over 300 other micros

Upgrades available from all previous versions
for only $25.00 Call for Authorization

To Order Contact:

XENOSOFT™

2210 Sixth Street, Berkeley, CA 94710
(415) 644-9366

Number 123 on the Reader Service Card

★ NOW AVAILABLE FOR C ★ QUICKBASIC ★
★ IBM BASIC/BASICA ★ TURBO PASCAL ★

The **A**spen **S**ystems **S**ubroutine **E**ditor (**ASE**) you can call from your programs. Design your own screen layouts - - update in several windows simultaneously. **ASE** is:

- ★ **TRANSPORTABLE** - can be configured for any keyboard and almost any computer or language under MSDOS.
- ★ **VERSATILE** - customize input for any application.
- ★ **COMPLETE** - with full screen edit capabilities and a wide variety of automatic conversions.
- ★ **EASY TO USE** - data and screen layouts described in a single map.
- ★ **FULLY DOCUMENTED** - including examples in BASIC, PASCAL, FORTRAN, C, and even COBOL.
- ★ **AFFORDABLE-**
  ASE is still only..... **$99**
  DEMO (See what ASE can do for you!)..... **$ 3**

The **A**spen **S**ystems **S**ubroutine **P**ackage (**ASP**) - the companion package for **ASE** - with many functions unavailable or difficult to perform in high level languages. Like **ASE, ASP** is:

TRANSPORTABLE ★ FULLY DOCUMENTED
EFFICIENT ★ AFFORDABLE

ASP includes 150+ subroutines: conversions; sorts; string, bit, date/time functions; decimal arithmetic, and **MORE**.

ASP is still only..... **$130**

ASE and **ASP** include support and demo programs.

Prices are PPD (continental USA).
Colorado Residents add 3%.

P. O. Box 1163
Grand Junction, CO 81502
(303) 245-3262
VISA/MasterCard accepted

ASPEN SYSTEMS

CP/M and MS-DOS are trademarks of Digital Research and Microsoft, respectively.

Number 86 on the Reader Service Card

Software Tools

For Programmers & Non-Programmers

★ NEW ★ VERSION

Opt-Tech Sort™

★ NEW ★ VERSION

Opt-Tech Sort is a high performance Sort/Merge/Select utility. It can read, sort and write a file faster than most programs can read the data. Example: 1,000 records of 80 bytes can be read sorted and a new file written in less than 9 seconds (IBM XT). Opt-Tech Sort can be used as a stand-alone program or Called as a subroutine to over 30 programming languages.

Numerous features are included. A partial list of features includes: The ability to process files of any size. Many filetypes are supported including Sequential, Random, delimited, Btrieve, dBASEII & III and many others. Up to 30 key fields can be specified over 16 different types of data are supported. Powerful record selection capability allows you to specify which records are to be included on your output. Record reformatting allows you to change the structure of your output record and to output special fields such as record numbers for use as indexes. **Newest Features: Faster than ever! LIM EMS support. User exits to pass or receive records. Less disk space required and much more!** MS-DOS $149.

On-Line Help™

On-Line Help allows you to easily add Help Windows to all your programs. On-Line Help is actually two help packages in one. You get BOTH Resident (pop-up) and Callable help systems.

The resident version allows you to add help to any system. Your Help system is activated when the "Hot Keys" that you specify are pressed. You can then chain between help windows in any manner you desire. Upon clearing the help windows you are right back into what ever program you left.

The callable version allows you to easily display help windows from your programs. A simple Call to the help system makes the window appear. On-Line Help is callable from over 20 different languages.

You have full control over the help window content, size, color and location. MS-DOS $149. Demo $10.

Scroll & Recall™

Scroll & Recall is a resident screen and keyboard enhancement. It allows you to conveniently scroll back through data that has gone off the top of your display screen. Up to 27 screens of data can be recalled or written to a disk file (great for documenting systems operations). Also allows you to easily recall and edit previously entered DOS commands without retyping.

Scroll & Recall is very easy to use. Its a resident utility that's always there when you need it. MS-DOS $69.

Visa, M/C, AMEX, Check, Money Order, COD or Purchase Orders accepted

To order or to receive additional information just call and receive immediate highly qualified attention!

Opt-Tech Data Processing
P.O. Box 678 / Zephyr Cove, NV 89448
(702) 588-3737

Number 16 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

lator manual. The Stroustrup book (327 pages) provides both a tutorial and a description of C++ by the language's designer. The Guidelines manual (105 pages) includes a few pages on installation and a tutorial, but otherwise consists wholly of the AT&T release notes for C++. The AT&T release notes are divided into three sections: (1) software features and notes, technical tips, and a language overview; (2) complex arithmetic in C++; and (3) a set of C++ Classes for Co-Routines (which are not implemented in Guidelines C++). Two AT&T addenda document changes between Versions 1.1 and 1.2. The manual has a table of contents, but no index. Example programs that correspond with chapters in the Stroustrup book are provided on disk. Although the Guidelines C++ package was ported specifically for Microsoft C under DOS, I was able to compile the translated files with both the MetaWare and the NDP 386 C compilers. Of course, to generate executable files, you would have to write equivalent C++ library functions to replace those provided in the Guidelines supplied libraries (typically 40K).

OS/2 & DOS C 5.1, Pascal 4.0, FORTRAN 4.1, & BASIC 6.0 compilers, MASM 5.1, and OS/2 Toolkit
Microsoft Corporation
P.O. Box 97017
Bellevue, WA 98073
(800) 426-9400

I can't do this powerhouse of new products justice in this short synopsis, so my comments will be brief. Microsoft has released six new language products for DOS and OS/2: updated development systems for C ($450), Pascal ($300), FORTRAN ($450), and BASIC (6.0); an updated version of MASM ($150); and an OS/2 Programmer's Toolkit ($350). I've had a chance to briefly review all but the Pascal compiler. The compilers and MASM are bundled with the new Microsoft editor, the linker, the librarian, the CodeView source code debugger, and other utilities. The Microsoft editor (called M in the real mode and MEP in the protected mode) is emacs-like and supports a somewhat configurable keyboard interface. All of these compilers and MASM are provided with versions that execute under both DOS and OS/2 and that support writing bound (dual mode) applications that run under both DOS and OS/2. Only C and MASM offer full support for OS/2, allowing multi-threaded OS/2 applications and dynamic linking. FORTRAN and BASIC support only major features of OS/2, including direct calls to the OS/2 application programmer's interface (API). To support multiple threads and dynamic linking, the runtime libraries of FORTRAN and BASIC must be rewritten.

Each language system comes on a number of 5.25-inch disks (360K for DOS and 1.2 megabyte for OS/2) with a SETUP program. Distribution is available on 3.5-inch disks if you return a coupon to Microsoft. The SETUP program is almost mandatory given the number of disks in each distribution and the complexity of two operating systems, multiple runtime libraries, different memory models, math chip support, and mixed language programming. The SETUP program allows you to choose to install a DOS version, an OS/2 version, or both versions of the language. In each distribution, some programs execute only under OS/2 (the protected mode CodeView and editor), some under DOS or in the OS/2 DOS compatibility box, and some under both DOS and OS/2 as bound (dual mode) applications. It took me about two hours and several tries to get the C compiler and utilities installed for both DOS and OS/2 the way I preferred it. The other packages are less complex and would require less time. The SETUP utility will by default use three different BIN subdirectories for each type of program. Some of these packages have slightly different versions of the same software—the C system's versions of several utilities were newer than MASM's, for instance.

For the most part, the actual compilers have only limited changes from the previous versions with the exception of generating OS/2 code and having OS/2 runtime libraries. For example, the C compiler has the same manual as C 5.0, but with some updates covering compiler changes (60 pages), the editor (120 pages), new or updated utilities, and OS/2 features (73 pages). Changes include two new command-line switches, some new ANSI C features, the addition of several new pragmas, a few new keywords to support DLL and configuring registers, a half dozen new runtime library routines, and a list of functions supported only in the Real Mode. As in the past, the documentation is very good if a bit on the voluminous side.

The language system documentation does not cover the OS/2 API and features, which are covered by the OS/2 Programmer Toolkit documentation. The OS/2 Toolkit contains several tools, documentation, and sample programs for developers writing OS/2 programs. The toolkit disks (distributed on both 5.25-inch and 3.5-inch disks) include example OS/2 programs, Peter Norton's Guide to OS/2 (an online help system called QuickHelp), and several OS/2 utilities, many of which are duplicated in the language system distributions. Documentation is in two manuals: (1) a *Learning Guide* (75 pages) and *Programming Tools* (70 pages); and (2) the *OS/2 Programmer's Reference* (580 pages). The OS/2 Toolkit documents the OS/2 API functions and I/O controls with short appendices on utility macros, screen devices, code pages, OS/2 executable file formats, and ANSI Escape sequences. If you were expecting the OS/2 Toolkit to function as an inexpensive version of the OS/2 Software Developer's Kit (SDK), you'll be disappointed. The Toolkit omits a number of OS/2 features (device drivers and the Presentation Mana-ger, for example) and therefore does not replace the much more expensive OS/2 SDK. ♦

The
Heap Expander™
version 2.0

Now your programs can have virtually unlimited heap space using expanded memory, extended memory, disk space, or any combination of the three. And it's all transparent. The Heap Expander's initialization code checks the system's resources and uses whatever is available.

still $59.95*

- Uses LIM-standard expanded memory if present.
- Uses AT-style extended memory if present.
- Swaps data to disk as needed.

Libraries and Source Code for:
- Turbo C
- Microsoft C 4.0 and 5.0
- Turbo Pascal 3.0 and 4.0

Requires an IBM PC, XT, AT, or close compatible with MS-DOS or PC-DOS version 2.0 or above

MC/VISA/COD call
1-800-248-1045 x 100 (US)
1-800-952-5560 x 100 (Idaho)

*Idaho residents add 5% sales tax
Foreign customers add $4.00 for shipping and handling.

The Tool Makers
75 Mt. Hermon Rd.
Scotts Valley, CA 95066
(408) 438-7412

Number 166 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LOW COST ADVERTISING

If you sell products to or provide services for professional programmers, a PJ Billboard ad is an easy, low cost way to reach more potential buyers. Regular listings include a headline, company name, address and phone number, Reader Service number and approximately 400 characters of sales pitch. Display listings are 1 5/8" wide by 2 3/8" deep. Costs are $125 for a single insertion, $115 per insertion for 3 insertions and $95 per insertion for 6 consecutive insertions. PJ Billboard ads must be prepaid and are non-commissionable. Call the ad department for deadlines.

Programmer's Journal
P.O. Box 30160
Eugene, OR 97403
503-747-0800

UNDERSTANDING C VIDEO TAPE

Get guaranteed results with this three volume video course! For expert or novice—these tapes (total 6 hours) cover ANSI C for today's professional programmer. Loaded with examples, these tapes cover it all! Handy reference book included.

$59.95 postpaid. Add appropriate tax in TX. Send check or money order to:

AUTOMATION EDUCATION
2309 Royce Drive
Arlington, TX 76016-1232
817-496-4125 **RS# 207**

PL/M TO C TRANSLATOR

LISP-based translator for major assistance in translating from INTEL PL/M to C language. Human readability a high priority. Sample listings available. Runs under MS-DOS 2 and up. Ports possible for VAX with VMS or BSD 4.2.

$5000 site license includes 3 months of telephone support.

Frank Korzeniewski Consulting
1564-A Fitzgerald Drive #137
Pinole, CA 94564
415-799-1819 **RS# 209**

MNEMATICS/PJ SPECIAL

Free subscription to the Mnematics Videotex service for ***PJ*** readers. Code listings from the current issue of ***PJ*** are now available for downloading. Circle reader service number **65** on the reader service card.

Mnematics Videotex
P. O. Box 19
Sparkhill, NY 10976-0019
800-322-3633
914-761-4700 in NY **RS# 65**

PERSONAL MEASURE

PC hardware and software performance measurement package. Use for system tuning, product evaluation and benchmarking. TSR less than 10K, easily disinstalled. Menu-driven analysis and reporting, graphically based and easy to understand. DOS 2.0+; CGA, EGA, VGA, Hercules or compatible graphics; FX-80, Proprinter, Laserjet or compatible printer.

$69.95, add $5 S&H, $5 for 3 1/2" media, 5% tax for Mass. residents. VISA/MC. Sensible 30 day money back guarantee.

Spirit of Performance, Inc.
73 Westcott Road
Harvard, MA 01451
617-456-3889
MCI Mail 357-1400 **RS# 197**

PROGRAM FASTER, INSTANTLY

EXPERT KEY automatically types variables and words at the touch of a single hot key. No macro keys to program or remember.

Memory resident program learns YOUR words from the keyboard and screen. Start typing and a highlighted word pops up. Hit the hot key and the word is completed or keep typing and the word is gone. Variable activity level. Program learns editing sequences and can recall previous command lines. Tracks keystrokes typed and saved. Runs with your editor, DOS and other applications.

LIMITED OFFER $65 (Reg. $95). Add $3 S&H. CA residents add 6%. For PC/XT/AT with 30 day guarantee.

Newell Technologies, Inc.
27758 Santa Margarita Pwky. #330C
Mission Viejo, CA 92691
714-583-7848 **RS# 203**

SEE CURRENT FILE ACTIVITY

WATCH I-O is a pop-up utility that displays files open, DX:DS of open and last read/write, last read/write counts, last Exec, last file opened, last file-not-found. User can select hot key, pop-up for file opens, pop-up for file-not found, track file opens on printer.

WATCH I-O is only $49.95. With anti-virus option $69.95. VISA/MC.

Quantum Publishing
23860 Miles Road
Cleveland, OH 44128
800-342-6237 **RS# 201**

PJ ISSUE SPECIAL!
See ad on page 76 for details

AT LAST! A C SOURCE CODE ANALYSIS TOOL WITH FULL PREPROCESSING CAPABILITY

C:LINES/C:TREE features

- ANSI draft compatible preprocessor
- Outlined listings showing flow of control within procedures
- Source code reformatter

$79.95, shipping $3 domestic, $15 overseas. MD residents add 5% sales tax. Requires PC compatible, 256K.

SoftRex
4807 Bethesda Ave. #287
Bethesda, MD 20814
301-881-8274 **RS# 185**

C-TREIVE ISAM LIBRARIES

Compatible with C-ISAM and X/OPEN standards.

- FAST!
- No run-time royalties.
- Part of a family of SQL data base products (inquiries invited).
- Supports transactions and journaling.
- Contains extensions to C-ISAM.
- Full telephone support provided.
- Excellent documentation.
- Attractively priced.

Coromandel Industries Inc.
P.O. Box 620
Gracie Station
New York, NY 10028
RS# 192

PJ CODE IS AVAILABLE!

On disk from:

Programmer's Journal
503-747-0800
See ad on page 76.

Mnematics Videotex
P. O. Box 19
Sparkhill, NY 10976-0019
800-322-3633
914-761-4700 in NY **RS# 65**

On BBS:

FIDO/OPUS Network
Blazer Net 152 Node 9
503-343-9724

Turbo Source Search
P.O. Box 876
Scituate, MA 02066
617-545-9131 **RS# 35**

Software Developers

We need your program!

Do you have a program that's good enough to sell, but don't want the problems or financial risk of producing, typesetting, printing, packaging, warehousing, marketing, distributing and supporting a product?

Why start your own software house?
We've done it for you!

Merlin Publishing Group
is now accepting submissions of micro-computer software for publication.

You get:
- initial cash payments
- generous royalties
- to spend your time programming

But we can't help you if you don't submit. Call or write today for our submission guideline kit.

MERLIN
PUBLISHING GROUP ™
1240 Johnsons Ferry Place, Suite A10
Marietta, GA 30068
(404) 977-6034

Number **141** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ADVERTISER INDEX

Reader Service#	Advertiser Name	Page Number
180	A.I. Architects, Inc.	21
184	American Cybernetics	55
191	Andsor Research, Inc.	45
162	Anji Software, Inc.	42
170	ASCII	16
86	Aspen Systems	81
30	Austin Code Works	17
207	Automation Education	83
51	Az-Tech Software, Inc.	51
3	Borland International	C4
38	C Ware Corporation	56
56	CompuView Products, Inc.	50
192	Coromandel Industries, Inc.	83
*	Creative Programming Consultants	64
8	Ecosoft, Inc.	14
29	Eigenware Technologies	80
173	Elan Computer Group, Inc.	73
113	Elfsoft	75
206	Flambeaux Software	54
118	Fleming Software	64
209	Frank Korzeniewski Consulting	83
135	Gilmore Systems	31
*	Gimpel Software	65
204	Guidelines Software	43
122	Hauppauge Computer Works, Inc.	47
154	HavenTree Software LTD	15
169	IGC	5
202	Javert, Inc.	75
205	*Journal of Object-Oriented Programming*	25
155	Komputerwerk	26
63	Laboratory Microsystems, Inc.	75
44	Lahey Computer Systems	51
188	LALR Research	39
141	Merlin Publishing Group	83
143	Metagraphics Software Corporation	24
45	MetaWare, Inc.	1
121	MISOSYS Inc.	72
65	Mnematics Videotex	83
94	MMC AD Systems	65
*	PLEASE CONTACT THESE ADVERTISERS DIRECTLY	

Reader Service#	Advertiser Name	Page Number
109	NanoSoft	71
203	Newell Technologies, Inc.	83
160	Nu-mega Technologies	3
16	Opt-Tech Data Processing	81
195	Paradigm Systems	48
145	Pathfinder Associates	46
200	PC Tech, Inc.	35
*	*Programmer's Journal* Mail Order	76
6	Periscope Company, The	79
200	Programmer's Paradise	7
199	Prospero Software, Inc.	10-11
189	QCAD Systems	54
97	Qualstar Corporation	80
201	Quantum Publishing	83
147	Quarterdeck Office Systems	C2
198	Research Group, The	69
174	Seidl Computer Engineering	44
148	SemWare	16
175	Sharpe Systems Corporation	57
186	Silver State Software	56
165	SLR Systems	49
33	Soft Advances	59
177	Soft Evolutions, Inc.	26
185	SoftRex	83
98	Software Family, The	58
115	Software Interphase	37
110	Softway, Inc.	51
136	Sophco, Inc.	27
197	Spirit of Performance, Inc.	83
158	Summit Information Systems, Inc.	74
85	Sunny Hill Software	78
171	Systemat Corporation	8
166	Tool Makers, The	82
25	TurboPower Software	73
35	Turbo Source Search	70
125	V-Communications	20
196	West Chester Group	48
123	Xenosoft	81
182	Zortech, Inc.	C3
*	PLEASE CONTACT THESE ADVERTISERS DIRECTLY	

When contacting any of the above advertisers, please tell them you saw their ad in ***Programmer's Journal*** and that you appreciate their continuing support of this magazine.

PJ ISSUE SPECIAL

BORN TO PROGRAM PACKAGE
Back Issue 4.6—"Born to Program"
& Born to Program T-Shirt
Regular $19.00 Value—
For a Limited Time Only

Special Price $14.95

Former Editor Robert Keller outdid himself on this one. Volume 4.6 is vintage, groundbreaking *PJ*: Technical Editor Steve Baker on undocumented TSR functions; George Defenbaugh on DOS functions 45H and 46H; Steve O'Brien and Wing Kit Cheung with assembler source code for DOS function calls from Turbo Pascal; James Smith on Turbo Pascal inline code; Hal Hardenbergh; Frank Greco; Paul Barkley on laser printers and Genifer; Peter Feldmann on Advanced Trace86; and more.

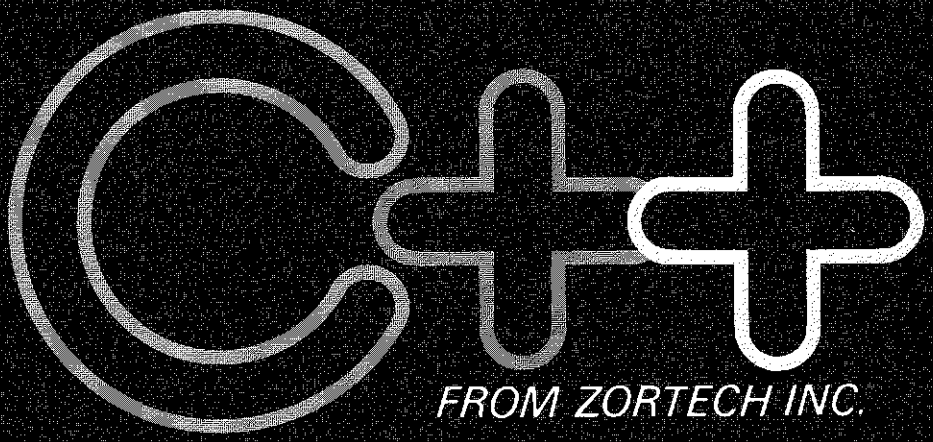
Born to program? Get your "colors" today!
See ad on page **76**, offer expires September 1, 1988

ADVERTISE IN

Programmer's Journal
Sept/Oct Issue, Volume 6.5
Space Reservation Deadline:
July 8, 1988
Materials Due:
July 22 , 1988
Call ***Programmer's Journal*** Ad Department to reserve your space today!

(503) 747-0800

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



The change to a pure language

Now, C programmers can move over to C++ with Zortech C++ – the world's first 'true' C++ compiler for MS-DOS machines.

Zortech C++ is a 'true' compiler and fully conforms to Bjarne Stroustrup's specification as outlined in his book 'The C++ Programming Language'.

Previous implementations of C++ were actually 'translators' – only able to translate C++ source code into C. Of course, this was unacceptable due to the long translating and compiling times.

Now, C++ comes of age with the introduction of the world's first true C++ compiler – from Zortech!

■ *Object Oriented Programming*

C++ is to C what Modula 2 is to Pascal. C++ brings 'classes' to C, so you can create separate modules that contain their own data and data-related operations. These 'classes' then become new types that can in turn be used to create further modules – this allows you to practically create your own language.

■ *ANSI C Superset*

You don't have to throw away your existing C programs – C++ is a superset of ANSI C. Now, you can take your Microsoft C or Turbo C compatible programs and easily migrate to C++ to take full advantage of the new C++ features.

■ *'Codeview' Compatible*

Zortech C++ is compatible with 'Codeview' – Microsoft's industry standard source code debugger.

■ *Improved Program Structure*

As stated in 'The C++ Programming Language', by using C++ "It would not be unreasonable for a single person to cope with 25,000 lines of code".

■ *Other benefits*

Here's just a few: Operator overloading, overloading function names, default arguments to functions and better type checking.

ESSENTIAL READING!

This 325 page book 'The C++ Programming Language' by Bjarne Stroustrup contains the original definition of C++. All the examples shown in this book have been successfully compiled and executed with the Zortech C++ Compiler.

ZORTECH

BOSTON · LONDON · FRANKFURT · GENEVA

Number 182 on the Reader Service Card

YES!
Rush me
C++ as shown
below:

☐ *Zortech C++* *$99.95*
☐ *C++ Book* *$29.95*

VISA/MC/COD/CHECK ACCEPTED

Name.....

Address.....

..... *Phone*.....

VISA or MC..... *Exp. Date*.....

To: ZORTECH INC. 361 Massachusetts Ave., Arlington, MA 02174.
Tel: 617-646-6703. Fax: 617-648-0603.

CALL THE ORDER HOTLINE 1-800-848-8408

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

"Behind the beauty of the Turbo C environment stands the brawn of a full-fledged compiler"

Marty Franz, PC Tech Journal

"Taking compilers and program development tools into the next generation is Borland International's Turbo C, a $99.95 package that will stun you with in-RAM compilations that operate at warp speed.

. . . a 21st century compiler at a preinflation 1967 price. Is it any wonder that Turbo C was included in the Best of 1987?

Richard Hale Shaw, PC Magazine

Turbo C represents an all-new price-performance level—one that will be hard to match, much less beat.

Stephen Randy Davis, PC Magazine

Turbo C showed excellent compiler speeds, good overall benchmark scores, and extraordinary floating-point performance."

Scott Robert Ladd, Micro Cornucopia

Actual photograph of Turbo C graphics displayed on IBM 8514 screen.*

Our new Turbo C 1.5 is a technological tour de force

At Borland we believe the slow way is no way, so Turbo C® is a racer. And as well as white-knuckle speed, Turbo C also gives you spectacular graphics.

Number 3 on theReader Service Card

Some of the reasons why the critics are so enthusiastic about Turbo C 1.5

Turbo C now includes:

- A professional-quality graphics library of over 70 functions
- A librarian that allows you to build your own object module libraries
- Context-sensitive help for the language and the library routines
- Text/video functions, including windows
- 43- and 50-line mode support
- VGA, CGA, EGA, Hercules, and IBM 8514 support
- File search utility (GREP)
- Sample graphics applications
- More than 100 new functions

The professional optimizing compiler for less than $100.00

For professional-quality C at a sane price, nothing comes close to Turbo C. It's super-fast and super-graphic. (We used it ourselves to write Eureka:™ The Solver and to develop the presentation-quality graphics in Quattro,™ our new and highly successful professional spreadsheet.) No one can deliver technical superiority like Borland.

*60-Day Money-back Guarantee***

For the dealer nearest you,
Call (800) 543-7543

BORLAND INTERNATIONAL

***Minimum system requirements:** For the IBM PS/2™ and the IBM® family of personal computers and all 100% compatibles. PC-DOS (MS-DOS®) 2.0 or later. 384K.
*Artwork metafile courtesy of Genigraphics® Corporation
**Customer satisfaction is our main concern. If within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department, and we will arrange a refund.

All Borland products are trademarks or registered trademarks of Borland International, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders. Copyright ©1987 Borland International, Inc. BI 1221

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED